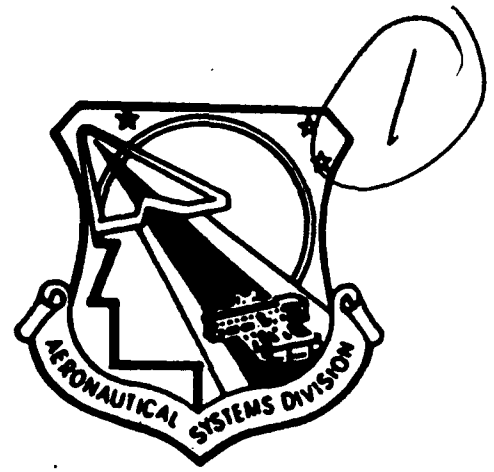


ASD-TR-92-5005

AD-A256 630

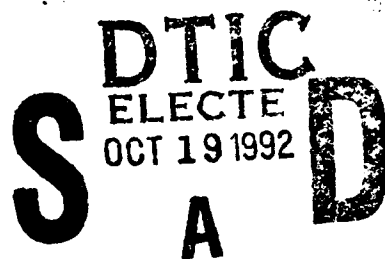


Systems Analysis Quality Metrics



William R. Williamson
Adroit Systems, Inc.
Dayton Technical Center
2970 Presidential Drive, Suite 340
Fairborn, OH 45324

December 1991



Final Report for Period May 1991 - November 1991

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.

DCS, Development Planning
Aeronautical Systems Division
Air Force Systems Command
Wright-Patterson AFB, OH 45433-6503

425 657

92-27387



8218

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

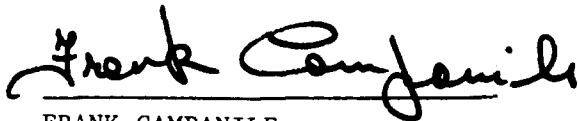
This technical report has been reviewed and is approved for publication.



ROBERT F. BACHERT
Project Manager
Development Planning Directorate



NANCY L. CLEMENTS
Chief, Requirements, Studies and Analysis
Development Planning Directorate



FRANK CAMPANILE
Chief, Requirements, Studies and Analysis
Development Planning Directorate

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify ASC/XRM, WPAFB, OH 45433-6503 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE December 1991		3. REPORT TYPE AND DATES COVERED Final Report: May 91-Nov 91	
4. TITLE AND SUBTITLE Systems Analysis Quality Metrics				5. FUNDING NUMBERS C-F33657-91-C-2148 PE-65502F	
6. AUTHOR(S) William R. Williamson					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Adroit Systems, Inc. Dayton Technical Center 2970 Presidential Drive, Suite 340 Fairborn, OH 45324				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Robert Bachert (513-255-6261) Advanced Systems Analysis DCS, Development Planning (ASD/XRM) Aeronautical Systems Division Wright-Patterson AFB, OH 45433-6503				10. SPONSORING/MONITORING AGENCY REPORT NUMBER ASD-TR-92-5005	
11. SUPPLEMENTARY NOTES This is a Small Business Innovative Research Program, Phase I					
12A. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited				12B. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) The objective of the Systems Analysis Quality Metrics program is the development of model quality metrics for the IDEF ₀ methodology in particular and for structured analysis methodologies in general. The previously existing IDEF ₀ model quality metrics are reviewed, critiqued and expanded. Additional metrics, not based on previous writings, are identified. This produces a useful model quality metrics baseline. Formalizations are recommended for specific IDEF ₀ elements which are not well enough defined to support metrics development. Techniques for the combination of metric primitives into whole diagram and whole model metrics are introduced.					
14. SUBJECT TERMS IDEF ₀ SADT ICAM Definition language Function Model Quality Metrics Formalizations				15. NUMBER OF PAGES 74	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	17. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL		

TABLE OF CONTENTS

Preface.....	vii
1.0 Executive Summary.....	1
1.1 Phase I Research Objectives and Results.....	1
1.1.1 IDEF ₀ Quality Metrics	1
1.1.2 Generalization.....	2
1.1.3 IDEF ₀ Formalization	2
1.1.4 Machine Detectable Quality Factors	2
1.1.5 Advanced Training Course.....	3
1.2 Model Utility, Model Validity, Model Quality	3
1.3 Organization of Final Report	4
2.0 Baseline.....	5
2.1 Syntax and Semantics.....	5
2.1.1 Syntax.....	5
2.1.1.1 Completeness.....	5
2.1.1.2 Correctness	6
2.1.1.2.1 Local Construction Syntax.....	6
2.1.1.2.2 Global Construction Syntax.....	8
2.1.1.2.3 Model Construction Syntax.....	9
2.1.2 Semantics	9
2.1.2.1 Completeness.....	9
2.1.2.2 Conciseness.....	10
2.1.2.3 Consistency.....	11
2.1.2.4 Correctness	12
2.1.2.5 Complexity/Understandability.....	13
2.1.3 Syntax and Semantics Summary	14
2.1.3.1 Syntactic Metrics.....	14
2.1.3.2 Semantics Metrics	14
2.2 Coupling and Cohesion.....	15
2.2.1 Coupling.....	16
2.2.1.1 Nature of the Connection	16
2.2.1.1.1 Pathological Coupling	16
2.2.1.1.2 Control Coupling	17
2.2.1.1.3 Normal Coupling	17
2.2.1.2 Structure of the Connection	17
2.2.1.2.1 Environmental Coupling.....	17
2.2.1.2.2 Record Coupling	18
2.2.1.2.3 Abstract Coupling.....	18
2.2.1.3 Combined Viewpoint	18
2.2.1.4 Coupling Summary	19
2.2.2 Cohesion	19
2.2.2.1 Types of Cohesion.....	19
2.2.2.1.1 Functional Cohesion.....	19
2.2.2.1.2 Sequential Cohesion.....	19
2.2.2.1.3 Mechanism Cohesion.....	20
2.2.2.1.4 Communicational Cohesion	20
2.2.2.1.5 Procedural Cohesion.....	22
2.2.2.1.6 Temporal Cohesion.....	22

2.2.2.1.7	Logical Cohesion.....	22
2.2.2.1.8	Coincidental Cohesion.....	24
2.2.2.2	Cohesion Metrics	24
2.2.2.3	Cohesion Summary	26
2.2.3	Coupling, Cohesion, and Complexity	26
2.2.4	Coupling and Cohesion Summary	27
3.0	Primitive Metrics	28
3.1	Syntax and Semantics.....	28
3.1.1	Diagrams	28
3.1.1.1	Context Diagrams	30
3.1.1.1.1	A-0 Diagram	30
3.1.1.1.2	Higher Level Context Diagrams.....	31
3.1.1.2	Decomposition Diagrams	31
3.1.2	Boxes and Arrows.....	31
3.1.2.1	Boxes.....	31
3.1.2.1.1	Box Syntax	31
3.1.2.1.2	Box Semantics	32
3.1.2.2	Arrows	33
3.1.2.2.1	Arrow Syntax	33
3.1.2.2.1.1	Required Arrows.....	33
3.1.2.2.1.2	Arrow Construction.....	34
3.1.2.2.1.3	Arrow Bundling, Branching and Joining.....	35
3.1.2.2.1.4	Arrow Tunnelling	35
3.1.2.2.2	Arrow Semantics	35
3.1.2.2.2.1	Bundling, Branching and Joining Semantics.....	35
3.1.2.2.2.2	Tunnelled Arrow Semantics.....	37
3.1.2.2.3	Rules for Drawing Arrows.....	38
3.1.3	Labels and Titles.....	40
3.1.3.1	Box Labels	40
3.1.3.2	Arrow Labels	41
3.1.3.3	Diagram Titles	45
3.1.4	Codes and Numbers	45
3.1.4.1	Box Numbering	45
3.1.4.2	Diagram Connectivity	45
3.1.4.3	Data Structure Connectivity.....	46
3.1.4.4	Reference Language.....	48
3.2	Decomposition Quality Factors.....	48
3.2.1	Coupling Metrics	48
3.2.1.1	Pathological Coupling	49
3.2.1.2	Control Coupling	49
3.2.1.3	Coupling Structure Metric	49
3.2.1.4	A Control on Every Box.....	50
3.2.2	Cohesion Metrics	51
3.2.2.1	Direct Interfaces.....	51
3.2.2.2	Information Integration.....	53
3.2.2.3	Inversion of Authority.....	54
3.2.3	Activations.....	55
3.2.3.1	Activation Theory	55
3.2.3.2	Activation Disclosure	57

3.2.3.3	Minimum Activations.....	58
3.2.3.4	Activation Language.....	58
3.2.4	Balance	58
3.2.4.1	Balance Between Box Detail and Arrow Detail on a Diagram....	58
3.2.4.2	Balanced Diagram Level of Detail.....	58
3.2.4.3	Balanced Level of Decomposition	59
3.2.4.4	Arrow Bundle Join and Branch Detail	59
4.0	Recommended Future Research	61
4.1	Formalizations.....	61
4.1.1	Arrow Labeling Requirements.....	62
4.1.2	Interpretation of Unlabeled Arrows	62
4.1.3	Data Content of Arrows	63
4.1.4	Mechanism Usage	63
4.1.5	Tunnelled Arrow Usage	64
4.1.6	Activation Definitions	64
4.1.7	Generic Interfaces	64
4.2	Algorithm Development	64
4.2.1	Syntax Metric Algorithm	65
4.2.2	Semantic Metrics Algorithm	65
4.2.3	Coupling and Cohesion Algorithms	65
4.2.4	Balance Algorithms	65
4.2.5	Activation Based Algorithms	65
4.2.6	Total Model Quality Score	66
4.3	Training Materials Development.....	66
4.4	Software Specifications	66

Accession For	
NTIS CRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability Codes	
Dis.	Avail and/or Special
A-1	

LIST OF FIGURES

Figure 1	Coupling Rankings	18
Figure 2	Functional Cohesion.....	19
Figure 3	Sequential Cohesion.....	20
Figure 4	Mechanism Cohesion.....	20
Figure 5	Communicational Cohesion	21
Figure 6	Interpretation of "Same" Data.....	21
Figure 7	Procedural Cohesion Example	22
Figure 8	Logical Cohesion, "Decomposition by Type"	23
Figure 9	Additive Functional Cohesion.....	24
Figure 10	For Comparison with Figure 9.....	25
Figure 11	Functional Cohesion Provided by Environmental Coupling.....	25
Figure 12	SADT Diagram Form	29
Figure 13	Meanings of the Sides of a Box	32
Figure 14	Precedence - Cohesion Inconsistency.....	33
Figure 15	Cyclic Processing	34
Figure 16	Connection of Open-Ended Boundary Arrows.....	36
Figure 17	Bundling for Clutter Reduction	36
Figure 18	Bundling to Depict Activation Logic.....	36
Figure 19	Inconsistent Joins and Branches	36
Figure 20	Conventions for Interpretation of Arrows at Branches and Joins	43
Figure 21	Arrow Segment Ambiguity	43
Figure 22	For Comparison with Figure 21.....	44
Figure 23	For Comparison with Figures 21 and 22.....	44
Figure 24	Fan-Outs.....	44
Figure 25	Implicit ICOM Code Assignment on Parent Diagram	46
Figure 26	ICOM Coding Example	47
Figure 27	Handling Tunnelled Arrow Connectivity.....	48
Figure 28	A Direct Interface Path.....	51
Figure 29	Two Direct Interface Paths	52
Figure 30	Maximum Number of Direct Interface Paths.....	52
Figure 31	A Broken Direct Interface Path	53
Figure 32	Environmental Coupling Source Off or On a Diagram	54
Figure 33	Box Activations and Truth Table.....	56
Figure 34	Decomposition of Box from Figure 33.....	57

PREFACE

This is the Final Report on Contract F33657-91-C-2148, Systems Analysis Quality Metrics, a Small Business Innovative Research (SBIR) Phase I contract. The contract was awarded to Adroit Systems, Inc., 809 N. Royal Street, Alexandria, Virginia, 22314 on 31 May 1991. The Principal Investigator was William R. Williamson. The work was performed at Adroit's Dayton Technical Center, 2970 Presidential Drive, Suite 340, Fairborn, OH 45324.

The sponsor of the research was the USAF Aeronautical Systems Division (ASD), Deputy for Development Planning (XR), Mission Area Planning (XRS) and Advanced Systems Analysis (XRM) Directorates. The XR project manager was Robert Bachert, who also contributed technical expertise to the project.

The research was completed on 30 November, 1991. An SBIR Phase II proposal is pending.

The research reported herein establishes feasibility of substantive, machine implementable, IDEF₀ quality metrics. A secondary objective was to stimulate additional metrics related research within the IDEF₀ community, and the funding thereof by supportive organizations. Many issues were resolved by this research. Several previously unpublished issues were raised. Feedback from the IDEF₀ community is solicited. Respond to Robert Bachert, ASD/XRM, Wright-Patterson AFB, OH, 45433 or to Bill Williamson at the above Adroit Dayton Technical Center address.

1.0 EXECUTIVE SUMMARY

Contract F33657-91-C-2148, Systems Analysis Quality Metrics, was awarded on 31 May, 1991 to Adroit Systems, Inc. The research effort, reported herein, was completed on 30 November, 1991. This was a Phase I Small Business Innovation Research (SBIR) contract. A Phase II proposal for continued research is pending.

1.1 *Phase I Research Objectives and Results*

The Phase I Systems Analysis Quality Metrics research project had five technical objectives:

- Develop a useful set of systems analysis quality metrics for a specific systems analysis methodology.
- Generalize the set of quality metrics to systems analysis methodologies in general.
- Formalize the specific systems analysis methodology in accordance with ASD/XR Premilestone I Planning criteria.
- Identify machine detectable quality factors for the specific systems analysis methodology.
- Develop advanced training course materials for the specific systems analysis methodology incorporating the quality factors and formalisms developed in the research program.

The accomplishment of these objectives in the Phase I effort is discussed in the following sections.

1.1.1 *IDEF₀ Quality Metrics*

The specific systems analysis methodology upon which the Phase I research was performed is the ICAM (Integrated Computer-Aided Manufacturing) Definition language for function modeling, IDEF₀.¹ Note: This is the document referred to throughout this report as the "Users Manual" (UM). IDEF₀ is the most widely used systems analysis methodology throughout the DoD and industry. The level of formalization of IDEF₀ which currently exists was adequate to support the Phase I research. However, several extensions to this formalization are recommended (Section 4.1).

There had been some nominal effort on the development of IDEF₀ quality metrics in the past. The results fell far short of being useful. These shortcomings were addressed in Mr. Williamson's paper to the May 1990 IDEF Users Group² and have been addressed in depth in the Phase I effort.

Previous writings on the measurement of IDEF₀ syntax provided a useful baseline. We recommend a greater level of specificity. The previous writings on IDEF₀ semantics contained useful material but fell short of providing a baseline. The Phase I effort extended the concepts into a useful semantics measurement baseline. Writings on

¹ "Integrated Computer-Aided Manufacturing (ICAM) Function Modeling Manual (IDEF₀)", UM110231100, Materials Laboratory, Air Force Wright Aeronautical Laboratories, Air Force Systems Command, Wright-Patterson AFB, OH 45433, June 1981

² Williamson, W. R., "Effective IDEF₀ Modeling --- Some Tricks of the Trade", presented at 1st 1990 IDEF Users Group, Washington, D.C., May 1990

coupling and cohesion defined a measurement scheme for primitives. This was a powerful and useful baseline. The techniques for combining the primitive measures into an overall measure for a whole diagram were found to be invalid. The development of a valid technique is proposed as part of the Phase II research.

Additional measures, not based on previous writings, were identified. These included the use of precedence relationships inherent in the methodology, the use of activation concepts as a measure of complexity, the measurement of clutter, and several balance concepts as measures of consistency.

The greatest shortcoming of previous writings was the lack of techniques for combining primitive measures into whole diagram and whole model measures. This spawned an Algorithm Development task proposed for the phase II effort.

1.1.2 *Generalization*

Most structured analysis methodologies are based on the same underlying concepts. Differences are in presentation, i.e., graphics, terminology, construction rules. Thus, it was anticipated that many of the quality factors developed for IDEF₀ would be generally applicable to other structured analysis tools.

Most of the general concepts relating to characterizations of model quality extend naturally to other structured analysis methodologies. The coupling and cohesion concepts are valid for any hierarchical decomposition based methodology. The use of activation scenarios as a measure of model complexity is applicable to most methodologies. The syntax and semantics metric concepts are generally applicable, although the specifics would, of course, be different for each methodology. The use of balance concepts as measures of consistency is valid for decomposition techniques.

1.1.3 *IDEF₀ Formalization*

It was predictable that as the quality metrics were developed, the need to define some IDEF₀ rules more precisely than they are presently defined in the UM would occur. This occurred more frequently than we had anticipated. We found the arrow labeling requirements to be loosely stated and that the conventions (from SADTTM) for the interpretation of unlabeled arrows were omitted from the Users Manual. The "All of the tokens" concept, which forms the basis for most of the coupling metrics, is not treated effectively in the UM. There are no well-defined rules for mechanisms, for the use of tunnelled arrows, nor for the enumeration of activations of boxes. There is no rule stating the inappropriateness of generic controls on boxes.

The development of formalizations was considered to be a by-product of the proposed Phase I effort. Due to the frequency of occurrence of ill-defined rules in authoritative documentation on IDEF₀, a Formalization task is required for the Phase II effort.

1.1.4 *Machine Detectable Quality Factors*

Generally, whether or not a quality factor is machine detectable depends on the extent to which natural language interpretation is required. For example, the cohesion concepts and the coupling structure concepts as currently described in the IDEF₀ Users Manual lend themselves well to machine interpretation. Most of the syntax and semantics metrics are machine detectable. These machine detectable factors form a rule base for an Expert System. The feasibility of Expert System development is well established by the large quantity of machine detectable quality factors identified in the Phase I effort.

TM SADT is a Trademark of SofTech, Inc

1.1.5 *Advanced Training Course*

The Phase I research significantly advances the state-of-the-art of IDEF₀ modeling. Mere insight into IDEF₀ quality factors improves one's expertise as a modeler.

The tutorial materials developed during the Phase I effort were presented at IDEF Users Group Workshops in Albuquerque³ and Ft. Worth⁴ and in the form of two-hour briefings to the IPO⁵, WRDC MANTECH⁶ and AFHLR/LRL⁷. Valuable feedback occurred on all occasions.

Although all courseware materials exist in the form of workshops and briefings, formal compilation into an advanced training course has not occurred. The formalization work must be completed before much of these materials have appropriate authorizations and approvals to be taught as advanced IDEF₀ techniques. Although largely a formality, Advanced Training Course development is proposed as a Phase II task.

1.2 *Model Utility, Model Validity, Model Quality*

"M is a model of S (a system) if M can be used to answer questions about S." Or, adding an element of practicality to the definition, **"M is a model of S if M can be used to answer (a well defined set of) questions about S (to a tolerance adequate for a stated purpose)."**⁸

The key issue in these definitions is model utility. Is the model useful? Model utility is comprised of two factors: model validity and model quality. In fact, if model validity could be quantified on a scale of 0 to 1, and if model quality were also quantified on a scale of 0 to 1, then the product of the two measures would be an excellent measure of model utility. This measure would effectively capture the fact that an invalid model is useless, regardless of its quality. The measure would exhibit the characteristic that the utility of a valid model is proportional to the quality of the model. A valid model might be useless if its quality is poor.

The purpose behind this discussion is two-fold. One purpose is to exclude model validity from the hypothetical list of model quality factors. Formally, the argument is that model validity takes precedence over model quality and therefore model validity cannot be considered to be a factor of (i.e., subordinate to) model quality.

³ Williamson, W.R., "IDEF Quality Metrology", Workshop presented at 1st 1991 IDEF Users Group, Albuquerque, NM, May, 1991

⁴ Williamson, W.R., "IDEF Quality Metrology", Workshop presented at IDEF Users Group Mini-Symposium, Ft. Worth, TX, October, 1991

⁵ Williamson, W.R., "IDEF₀ Quality Metrics", presented to Dictionary Methodology Committee and Application Validation Methodology Committee at the IGES (Initial Graphics Exchange Specification)/PDES (Product Data Exchange using STEP) Organization (IPO) Quarterly Meeting, Pittsburgh, PA, July, 1991

⁶ Williamson, W.R., "Systems Analysis Quality Metrics", presented to the Aeronautical Systems Division (ASD), Wright Research and Development Center (WRDC), Manufacturing Technology Directorate (MANTECH), Wright-Patterson AFB, OH, August, 1991

⁷ Williamson, W.R., "Systems Analysis Quality Metrics," presented to Air Force Human Resources Laboratory (AFHRL), Logistics and Human Factors Division (LRL), Wright-Patterson AFB, OH, November, 1991

⁸ Ross, D.T., "Removing the Limitations of Natural Language", Summary of a lecture presented at the Software Engineering Workshop, Schenectady, NY, 31 May 1979, SofTech internal document no. 9061-25, SofTech, Inc., 460 Totten Pond Rd, Waltham, MA, 02154, July 1979

The second purpose is to formally include, in model quality, all model utility factors which are not part of model validity. In other words, we're saying that model utility is comprised of exactly two factors: validity and quality. Factors such as satisfaction of purpose, specificity, level of detail, expressiveness, as well as syntactic and semantic compliance are all considered to be quality issues.

The purpose of this research is to demonstrate that all such quality factors are quantifiable and that, therefore, the total concept of model quality is quantifiable.

1.3 *Organization of Final Report*

Section 2 of this report, **Baseline**, is a critical review of the UM material on IDEF₀ model quality. This UM material qualifies as the baseline on the subject prior to the research reported herein. Section 3, **Primitive Metrics**, presents the results of our research into primitive metrics. We reorganize and elaborate on that portion of the baseline which we consider to be useful. We also introduce several primitive metrics not covered by the baseline material. We discuss required formalisms, machine detectability, and offer our preliminary thoughts on whole diagram and whole model metrics. Section 4, **Recommended Future Research**, is our recommendation on where we should go from here. The key topics are required formalizations and the development of algorithms for whole diagram and whole model metrics.

2.0 BASELINE

Section 6.6 of the UM is titled, "Model Quality Checklist". There are six subsections titled: "Syntax", "Semantics", "Relationship Between Coupling and Cohesion", "Metrics Based on Coupling and Cohesion", "Measures and Types of Cohesion", and "Assessing Coupling and Cohesion in IDEF". About half of the material addresses Syntax and Semantics and about half of the material addresses Coupling and Cohesion. This material was reviewed in detail to extract the substantive model quality baseline. The review is presented in the same sequence as the presentation of the material in the UM. Items in boldface type are UM quotations.

2.1 *Syntax and Semantics*

Syntax refers to the lexicon (symbols) and grammar (presentation rules) of a language. Semantics refers to the meanings of the lexicon and grammar of the language. The UM offers a section on syntax followed by a section on semantics.

2.1.1 *Syntax (UM 6.6.1)*

The author defines three categories of syntax: Local Construction Syntax --- referring to first-order combinations of primitives, Global Construction Syntax --- referring to whole diagrams, and Model Construction Syntax --- referring to whole models.

We noted one misleading statement in the definitions of the categories. The author states that first order primitives are defined for FEO's (and diagrams). An FEO is defined in the UM Glossary as "**A diagram ... in which violation of normal syntactic rules is allowed**". The important issue here is that quality metrics developed for FEO's should not consider syntactic violations to be shortcomings.

The author lists two standards for evaluation of syntax: Completeness and Correctness.

2.1.1.1 *Completeness*

The author defines completeness as the "**Degree to which all field-entries, labels, boxes, arrows, and identifying notations are present in a diagram or model.**" We'll address the listed items individually.

- **Degree to which all required field-entries are present ...**

The field-entries are the entries in the fields of the various IDEF₀ forms. Thus, this is simply a measure of whether or not the author(s) of a model filled out the forms. It is worth noting that some fields, e.g., node number, are more important than others. A fine tuned scoring metric should take this into account.

- **Degree to which all required labels are present ...**

Labels refer to the verb phrases in boxes and the noun phrases associated with arrows. Every box is required to have a label. Every arrow is required to have a label. (Note: Some SADT writings define conventions for omitting labels from some arrows emanating from branches and entering joins. These conventions were not included in the UM. Probably an oversight.)

- **Degree to which all required boxes are present ...**

Syntactically speaking, what are the **required boxes** which must be present? Obviously we wouldn't have much of a model without any boxes.

But, which ones are **required**? We can think of only two rules which state a requirement for a box. There has to be one on the A-O diagram. And, any diagram with two boxes is **required** to have (at least) one more box. Neither of these supports the need for a metric to measure the presence of **required** boxes.

- **Degree to which all required arrows are present ...**

There are several rules for required arrows in a model. Every box must have a control. Every box must have an output. A child diagram must include all of the arrows entering and leaving the parent box.

- **Degree to which all required identifying notations are present ...**

Identifying notations is assumed to include the items referred to in UM Section 2.2 as Diagram Interface Connectivity and Data Structure Connectivity, referred to in Section 3.2.1 as Reference Expressions, referred to in Section 6.5.2 as Reference Language, and discussed in Section 6.6.1.3 (without giving them another name).

In summary, relative to completeness, we would like metrics measuring the presence of required field entries, required labels, required arrows, and required reference syntax. We have dropped required boxes from the list. We will have specific metrics for the A-O diagram, which will include the requirement for a box. The "3 to 6 boxes per diagram" rule isn't really a required box rule. It will be handled as a Correctness rule.

2.1.1.2 *Correctness*

The author defines correctness as the "Degree to which a diagram or model has been accurately constructed, labeled, and identified; i.e., degree to which syntactic relations have been represented appropriately in the graphics, and constraints obeyed." This definition is followed by a list of seven Local Construction Syntax constraints, a list of four Global Construction Syntax constraints, and a page-and-a-half of text on Model Construction Syntax. We'll address these item by item.

2.1.1.2.1 *Local Construction Syntax (UM 6.6.1.1)*

a. One way arrow segments are made up of ordered pairs of endpoints (SOURCE, SINK), where:

SOURCE points are one of:

- boundary endpoint near diagram I, C, or M boundary
- box endpoint on Box O side
- fork sink
- join sink

SINK points are one of:

- boundary endpoint near diagram O boundary
- box endpoint on box I, C, M side
- join source
- fork source

Note that all combinations are legal except a (diagram boundary, diagram boundary) arrow.

Although technically correct, the attempt at formality here to define arrow segments (as distinct from whole arrows) will confuse many readers. It is objectionable to use the words "source" and "sink" in the definitions of the words "source" and "sink". Since you have to look beyond a fork or join, to the other end of the arrow, in order to determine

whether the fork or join is a source or sink, you might as well write the rule for whole arrows in the first place. In short, writing the rule for arrow segments hasn't brought you anything except the addition of complexity to what is basically a very simple concept.

The rule should simply say, "The source of an arrow is always the Output side of a box. The sink of an arrow is always the Control, Input, or Mechanism side of a box." That's the local construction version of the rule. Mechanism Calls are an exception, to both the rule stated here and to the arrow segment rules stated in the UM.

The global construction rule is simply a corollary to the local construction rule, "Since the source box and sink box of an arrow may not be on the same diagram, the boundaries of a diagram may be, as an artifice, treated as arrow sources and sinks. The Control, Input and Mechanism boundaries of a diagram are treated as allowable arrow sources. The Output boundary of a diagram is treated as an allowable arrow sink."

At the model construction level the special rules for tunneling and for the A-O diagram would be introduced.

The metrics for arrow source and sink syntax will be based on whole arrow definitions.

b. Arrow labels are associated with arrow segments. Line squiggles may be used to clarify the association.

The rationale behind the attempt to define arrow segments comes from this rule. The issue is that every arrow segment needs a label. Direct interface arrows (from one box to another on a diagram) require labels. Boundary arrows (from an I, C, M boundary to a box, from a box to an O boundary) require labels. The arrows entering a join and the bundled arrow leaving the join require labels. The bundled arrow arriving at a branch and the arrows leaving the branch require labels.

Note that this is completeness rather than correctness. In Section 2.1.1.1 we had **Degree to which all required labels are present ...**. What we have here, under the heading of correctness, is a more detailed description of the labels which must be present on arrows. The correctness issue has been overlooked --- "the label must be a noun, or noun phrase."

c. Each ICOM code from the parent box must be associated with the boundary endpoint of some arrow.

Technically, this is an incorrect statement. Figure 3-21 of the UM indicates that ICOM code assignments, implicit on the parent diagram, include tunnelled arrows. Thus all ICOM codes implicit from the parent diagram may not appear on the child diagram.

"A boundary arrow on a diagram has an ICOM code, placed near the boundary endpoint of the arrow if, and only if, the arrow also appears on the parent diagram." The format of the ICOM code is also a correctness issue. "The ICOM code consists of an alphabetic character; I, C, O or M; and an integer."

d. Boxes can have an integer number. Box numbers start at 1 and increase by 1.

The requirement for and placement of a number is local syntax. The numbering scheme is global. The local syntax rule is, "Every box must be identified by an integer number located in the lower right corner of the box."

The integer in the lower right corner of a box is another "identifying notation" which must be "present" --- completeness, not correctness. If one really wants to

split hairs, the requirement for the notation is completeness, and the specification that it should be an integer and where it should be located, "lower right corner," is correctness.

e. Names are associated with boxes and cannot stand alone.

"Every box must be labeled with a verb phrase placed inside the box."

The requirement to have a label is completeness. Its characterization as a verb phrase and its placement, "inside the box" is correctness.

f. Endpoints of arrows at diagram boundaries have tunnel "(" or an ICOM but not both.

This is mostly a completeness statement, the requirement for the presence of an ICOM code or parenthesis. "But not both" is correctness. We require a correctness statement for parentheses, "A boundary arrow on a diagram is enclosed in parentheses near the boundary endpoint of the arrow if, and only if, the arrow does not appear on the parent diagram."

g. Endpoints of arrows at box endpoints can have tunnels "(".

"Endpoints of arrows at sides of boxes must be enclosed in parentheses if the arrow does not appear on the child diagram."

2.1.1.2.2 Global Construction Syntax (UM 6.6.1.2)

- **A finished IDEF diagram consists of no less than three and no more than six boxes, unless it is an A-O diagram in which case it contains exactly one large box.**

Our metric will not consider the largeness of the box on the A-O diagram.

- **Each box must be named and numbered. On each diagram the numbers start at one and increase monotonically by one. The box on an A-O diagram is numbered zero. No two boxes on one diagram can have the same number.**

The requirement for labeling and numbering boxes was already stated in items d. and e. under Local Construction Syntax. "The boxes on an IDEF₀ diagram are consecutively numbered from 1 to n, where n is the number of boxes on the diagram. The number of the box on the A-O diagram is zero, whether or not the number is explicitly shown in the bottom right corner (the number may be omitted)."

- **Each box must have at least one arrow source at its O (right) side and one arrow sink at its C (top) side.**

These are Local Construction Syntax and important enough to be stated separately. "Each box must have (at least) one Output arrow." "Each box must have (at least) one Control arrow."

- **All endpoints of arrow segments at diagram boundaries must have either an ICOM code or tunnel "(", except on an A-O diagram for which there are neither ICOM codes nor tunnel "(".**

This was already stated in items c. and f. under Local Construction Syntax. If and when the UM is rewritten, the uniqueness of the A-O diagram should be described independently, prior to introducing the syntax for all other diagrams, so that the frequent statements of exception can be eliminated.

So, there were only four items under Global Construction Syntax, two of which were restatements of items under Local Construction Syntax. Actually, there are many more Global Construction considerations. But, each must be evaluated relative to

the issue of syntax vs good practice. Are they rules, or not? The UM doesn't make the distinction. The distinction is critical to the development of metrics. All of the material in Sections 6.3, 6.4, and the layout refinement part of Section 6.6.2 of the UM is syntactic in nature. Some of that material is rules. Some is just good practice recommendations. We present this material in Section 3.1.2.2 of this report.

2.1.1.2.3 *Model Construction Syntax (UM 6.6.1.3)*

The author chose not to use a list of constraints format for this section of the UM. The material describes Node Numbering for diagrams, including FEO's, Text, and Glossary, and a brief introductory-type discussion of ICOM codes. A comprehensive discussion would include material which is scattered throughout the UM, notably: Section 3.2.1, Section 3.2.3, Section 4.1, and Section 6.5.2. We present this material in Sections 3.1.3 and 3.1.4 of this report.

2.1.2 *Semantics (UM 6.6.2)*

The UM discusses five criteria for the measurement of model quality from a semantics perspective: Completeness, Conciseness, Consistency, Correctness, Complexity/Understandability.

2.1.2.1 *Completeness*

The author defines completeness as the "**sufficiency of information content to cover subject matter (context)**". Measures and parameters of completeness are stated for diagrams and for whole models (the author chose not to use the "global" and "model" terminology introduced in the Syntax section of the UM.):

"Diagram Level: For any diagram, except the topmost-Degree of adequate coverage of information with respect to the bounded context on the parent."

This is a restatement of the definition using different words. It offers neither measures nor parameters. Technically speaking, IDEF₀ has built-in completeness. If the inputs on the parent diagram have been transformed into the outputs on the parent diagram, constrained by the controls on the parent diagram, and utilizing the mechanisms on the parent diagram, then the child diagram is complete insofar as bounded context is concerned. The quality issue is amplification, or expanded detail. Does the child diagram tell me more than I already knew from the parent diagram? The difficult measurement parameters are embodied in the words "sufficient" or "adequate".

"Whole Model Level:

Degree of decomposition of boxes.

[Varies according to purpose of model; amount of detailing needed to fully communicate the purpose]"

Degree of detailing arrows from top to lowest level in a model

[Varies with level of detail and abstraction of function boxes]"

Here are some of the words we were looking for relative to the diagram level.

We would like to measure the degree of decomposition (amplification) present in the child of any box. We would like to measure the degree of decomposition detailed in the child diagram of each of the arrows associated with any box. These are both global level measures of amplification. At the model level we're interested in evaluating whether the number of levels of decomposition presented in the whole model is sufficient to satisfy the model's purpose.

In the text material preceding the checklist item the author briefly discusses amplification and balance as measures of completeness. The tradeoff between

amplification and complexity is noted. We would like to measure a "just right" amount of amplification. Not too much, not too little. We'll revisit this issue later, after discussing coupling and cohesion.

Balance is a consistency measure rather than a completeness measure. One can discuss balance at the global level --- balance between activity detail and arrow detail on a diagram. One can also discuss balance at the model level --- balance in amplification or complexity between diagrams at the same level and balanced levels of decomposition in different parts of a model. We elaborate on balance in the consistency discussion (Section 2.1.2.3).

2.1.2.2 *Conciseness*

The author defines conciseness as the **"Precision of information contained in and/or conveyed by a diagram or model; appropriateness of terminology and symbology. Absence of information peripheral to model's orientation."**

Precision and conciseness aren't quite the same thing. Conciseness refers to the achievement of brevity without loss of precision. It's the attempt to be both brief and precise. The difficulty in achieving conciseness in labeling boxes and arrows is the primary motivation for glossaries. You can use brief labels on diagrams, to avoid clutter, and provide the precision on a separate page in the glossary.

Appropriateness of terminology and symbology is correctness, not conciseness.

Absence of peripheral information is, of course, exactly what conciseness is all about.

The measures and parameters of conciseness are stated to be:

- **Information value of labels, titles, and function names**
- **Degree of redundancies in label names**
- **Naturalness of terminology: to fit --**
 - **Subject matter**
 - **Audience"**

It would be difficult to find a better measure of conciseness than the information value of labels. Unfortunately, we don't know how to measure the information value of labels. Degree of redundancy in labels is simply one, of several, characteristics related to information value. The author recommends the use of a glossary. We concur.

Naturalness of terminology has little, or nothing, to do with conciseness. There is an indirect relationship in that jargon-of-the-trade, including acronyms and abbreviations, is typically briefer than the more generic terminology for the same concept. For example, labeling an arrow "SOW" rather than "Statement of Work", for an audience familiar with the term.

In summary we find little substantive information in the UM discussion on conciseness which would support the definition of a metric. The recommendation to provide a glossary is good advice. Brevity is easily measured. One could compute the average number of words per label. This would provide a relative measure for the comparison of models. An absolute measure based on some standard --- optimum number of words per label --- is conceivable, but probably not worth the trouble it would be trying to get agreement on the standard.

It is questionable whether conciseness is a useful measure of model quality. We are concerned with clutter. An uncluttered model is probably sufficiently concise. A

cluttered model is not concise. But, boxes and arrows are as great a source of clutter as their labels. The best solution is usually a revised decomposition. We believe the coupling and cohesion metrics will adequately address clutter and that conciseness will occur as a natural by-product of clutter reduction.

2.1.2.3 Consistency

The author defines consistency from two perspectives, internal and external:

- "• **Internal: Degree of uniformity of notation, symbology, and terminology (within the model)**
- **External: Degree that content is traceable to the system being modeled (outside the diagram or model)"**

As was the case with conciseness, we have an element of buzz wordiness --- statements which sound OK when read casually, but which don't stand up to scrutiny.

Uniformity of notation and symbology is built into IDEF₀ in the first place. Shortcomings would be syntactic correctness problems. If one correctly uses the IDEF₀ syntax, uniformity of notation and symbology is automatic. This is not a consistency issue.

Uniformity of terminology is a consistency issue since terminology is not controlled by the syntax. However, like degree of redundancy and information value of labels in the conciseness discussion, uniformity of terminology is not measurable in a quantitative sense.

Traceability of model content to the system being modeled is a model validity issue, not a consistency issue. In particular, traceability is a measure of the ease with which a system expert can map the model onto the system. Model quality contributes to traceability. The system expert will find it easier to determine whether such a mapping exists if the model is of high quality. The use of natural language, specifically the system expert's language, in the model contributes to traceability. But, there is no provision in the model syntax and semantics for an audit trail from the model to the source system description.

The key issue is not whether the IDEF₀ model syntax and semantics support traceability but the fact that the IDEF₀ methodology supports traceability very effectively. The audit trail is embodied in the **author-reviewer cycle** comments. This key model quality indicator, traceability to the system, is not inherent in the final delivered model. It exists in the author-reviewer cycle kits prepared during model development.

The stated measures and parameters of model consistency are:

- "• **Correctness of ICOM labels: Placement and traceability through the hierarchy**
- **Correctness of tunnel () use**"

The author uses the word "correctness" and that's exactly what they are --- correctness measures, not consistency measures. Incidentally, the UM contains no rules for "correct" tunnel use. This is discussed in Section 3.1.2.2.2.2.

As was the case with conciseness, we find no substantive information in the UM discussion on consistency which would support the definition of a metric. However, we noted some consistency issues in our review of the UM discussion on semantic completeness. There are at least four identifiable balance concepts worthy of consideration under the heading of consistency.

- **Balance between activity detail and arrow detail on a diagram - "A diagram with very general and abstractly-labeled function boxes,**

but with minutely detailed labels on the data arrows is **unbalanced semantically**". (from UM discussion of semantic completeness on p. 6-29). Conversely, a diagram with explicitly-labeled boxes and abstractly labeled arrows is semantically unbalanced. Interpretation of labels is not easily captured in the form of a quantitative measure. However, the presence of an unbalance between activity and arrow detail on a diagram will become apparent in the decomposition of the diagram. We support this claim in Section 3.2.4.1.

- Balance between level of detail on diagrams at the same level of a model - Unbalance occurs, typically, in structured modeling endeavors in which different modelers are developing different parts of the model. One author does a better job of gradually introducing detail than another author. This unbalance is quantifiable, by comparing the number of boxes per diagram and number of arrows per box on diagrams at the same level of decomposition in different parts of the model.
- Balance between levels of decomposition in different parts of a model - Only relatively extreme occurrences of level-of-decomposition unbalance should be considered to be quality shortcomings. After all, one part of the system being modeled may be more complex than another part. But, extreme cases usually reflect a purpose and context problem. The author really only wanted to model part of the system. Level-of-decomposition unbalance is easily quantified by node analysis.
- Balance between arrow detail of joins and branches of an arrow bundle - The level of detail of arrows joining to create an arrow bundle and the level of detail of arrows branching from the same bundle should be similar. Ideally, the arrows forming a bundle and branching out of the bundle should be identically the same data items. This is measurable via interface analysis techniques, two-way tracing of the bundle to its sources and sinks and comparison of the data items at the two ends.

2.1.2.4

Correctness

The UM defines semantic correctness as the:

- **Extent to which information expressed in a diagram or model is an accurate description of the system being modeled.**
- **Degree to which implied constraint relations and data-to-function interfaces represent valid relations".**

The first bullet is model validity, not model quality (see discussion in Section 1.2). The second bullet can be interpreted generically, i.e.; whether arrows entering the left side of a box are, in fact, inputs; whether arrows entering the top of a box are, in fact, controls; whether arrows entering the bottom of a box are, in fact, mechanisms. These are sometimes difficult decisions and difficult to measure. Formalization issues are addressed in Section 3.1.2.1.2.

We noted in the consistency discussion that correctness of ICOM labels belongs here, under correctness. This is easily measurable and automatable. We noted that correct use of tunnelling is also a correctness issue, but that the UM contains no rules for correct tunnel use.

The UM offers as measures and parameters;

"• Review of model by experts

- Checks in company literature or textbook sources for terminology and technical usage."**

This is obviously good advice, but not measures nor parameters. As the UM points out, many semantic correctness items are subjective and difficult to quantify.

2.1.2.5 Complexity/ Understandability

The UM refers to complexity and understandability as dual measures. The UM definition (of understandability) is:

- "• Extent to which purpose of model or diagram is clear to the evaluator.**
- Degree to which "What the model/diagram says" is accurately depicted via the syntax."**

The purpose of the model is stated on the A-O diagram. We can't measure how clear that is to the evaluator. The degree to which something is accurately depicted has nothing to do with complexity. An accurate diagram might be very complex and a very simple diagram might be inaccurate.

The UM measures and parameters are:

"• Ease of finding the "main path"

- Number of activations per box**
- Number of potential scenarios**
- Facility of identifying activations and scenarios**
- Ease and success of "refining the layout"**
- Coupling and cohesion rating scheme"**

The main path in a diagram is an intuitive concept. And one must consider the possibility that the main path from a constraint perspective and the main path from a sequence or flow perspective may not be the same path. "Path" is a flow concept. Flow is not implicit in an IDEF₀ diagram. But, ignoring such philosophical issues, most people have an intuitive concept of a main path. On simple diagrams with three boxes and five or six arrows you can generally spot a path that qualifies. On busy diagrams with six boxes and 75 arrows you will have trouble finding a main path. That's really the issue --- How "busy" is the diagram? The worst contributors to busy diagrams are lots of arrows, particularly an abundance of feedback arrows depicting mutual constraints.

There is not a clear distinction in the literature between activations and scenarios. The terms are frequently used interchangeably. Regardless of the specific terminology, the point is that the arrival of data on the arrows associated with a box in different sequences and in different combinations leads to activations of the box at different times for different purposes. SADT included notations for the definition of activation rules for boxes as an optional part of the methodology. IDEF₀ did not adopt the SADT activation language. Nevertheless, the number of activations of a box is related to the number of

arrows associated with the box. The relationship isn't linear. Marca⁹ states that the number of possible activations is 2^n , where n is the total number of input, control and output arrows on the box. The binary base comes from the two states, presence or absence, of data on an arrow. A candidate metric for diagram complexity would be to base the metric on the number of possible activations per box.

"Ease and success of refining the layout" is pretty much buzz-words. When we're evaluating a delivered model we have no idea how easy it was for the author(s) of the model to refine the layout. To the extent that rules exist for layout, we are evaluating syntax rather than semantics. Layout is syntactic correctness, not semantic complexity. The UM offered five "rules for refining the layout of a diagram". In fact, two were layout rules and the other three were rules for labeling arrows. All are syntax rules.

The coupling and cohesion rating scheme provides valid measures of model complexity. Coupling and cohesion is discussed in depth in the following sections.

2.1.3 *Syntax and Semantics Summary*

Generally, the syntax and semantics material in the UM tends to be buzzwordy, incomplete and non-specific. It is, nevertheless, useful. It amounts to an overview of the subject. When viewed as an introductory first-cut at concepts which could logically lead to the development of substantive metrics the material is adequate. There is substantial "intelligent grunt-work" required to produce quantitative metrics.

2.1.3.1 *Syntactic Metrics*

The syntax quality factors were stated to be completeness and correctness. Completeness referred to the presence of required syntax in the model. Correctness referred to whether that syntax was appropriately applied.

The completeness quality indicators were the presence of required field entries, labels, boxes, arrows and notations. We noted that there are no specific requirements for boxes. We elaborated on required arrows as: 1) A control on every box, 2) An output on every box, and 3) Child edge-of-diagram boundary arrows inherited from parent box boundary arrows. We noted that a list of required notations would have to be compiled from several other sections of the UM. We do this in Section 3.1.2.4.

The correctness factors were categorized into local, global and (whole) model sets. Most of the local factors were completeness rather than correctness: requirement for every arrow segment to have a label, requirements for arrows to have ICOM codes or parentheses, requirements for boxes to have names and numbers. Under global factors we have: the 3 to 6 boxes per diagram rule, the every box must have a control rule, the every box must have an output rule, and the requirement to number the boxes sequentially --- all valid factors and all quantifiable. We noted that several additional factors exist in other sections of the UM which need to be compiled and evaluated from the perspective of rules vs good practice. Model factors included node numbering and ICOM coding. The ICOM coding material requires some expansion, from other UM sections. Presented in Section 3.1.2.4.3.

2.1.3.2 *Semantics Metrics*

Five quality factors were identified for semantics: Completeness, Conciseness, Consistency, Correctness and Complexity/Understandability.

⁹ Marca, David A. and McGowan, Clement L., SADT, Structured Analysis and Design Technique, McGraw-Hill Book Company, New York, 1988

Completeness was discussed from a global perspective and from a model perspective. Global completeness is pretty much built-in to IDEF₀ in the first place. The real issue is amplification. Model completeness is best measured in terms of amplification. Balance was discussed in the UM material. But, we felt that balance was consistency rather than completeness and discussed it from that perspective.

The UM discussion on conciseness focused on labeling. We believe the real issue is clutter reduction and that conciseness (of diagrams) occurs as a natural by-product of clutter reduction. Labeling conciseness is best achieved by the use of glossaries.

Consistency was characterized in the UM as uniformity of syntax and terminology, and traceability to the system being modeled. The measures were stated to be correctness of ICOM and parentheses use. The stated measures are, of course, correctness, not consistency. Traceability to the system being modeled is not inherent in the model. Traceability is, however, inherent in the author-reader cycle materials produced during model development. Uniformity of syntax is built-in to the methodology in the first place. Our discussion of consistency focused on balance measures.

Correctness of a model is a validity issue, as distinct from a quality issue. Correctness of ICOM coding and use of parentheses are valid correctness quality indicators. However, we noted that there are no rules for the "correct" use of tunnelled arrows.

Complexity and understandability are important quality factors. We disagreed with equating understandability to accuracy. We feel that complexity is best measured in terms of clutter and activations. Terminology such as "activations" and "scenarios" needs to be defined in the UM. The layout refinement rules offered are part of syntactic correctness --- part of the "several additional factors" referred to in Section 2.1.3.1.

2.2 Coupling and Cohesion

Coupling describes the degree of interconnectedness among components. Cohesion describes the degree of relatedness of subparts within a given part. (UM 6.6.3)

The distinction between interconnectedness and relatedness is subtle. Some authors consider the two concepts to be duals. We note that "components" and "subparts within a given part" are the same thing. If we restate the two characterizations as: "Coupling describes the degree of interconnectedness among activities on a diagram." and "Cohesion describes the degree of relatedness between the activities on a diagram." And, if we further note that cohesion is measured by evaluating the connections (arrows) between activities on a diagram. Then, we must conclude that that's about as subtle as you can get. We will elaborate as kernels of information occur in the review of the UM coupling and cohesion material.

Coupling basically refers to the extent to which one activity is dependent upon another as the result of something other than the explicit transfer of information. Cohesion refers to the extent to which one activity is dependent upon another as the result of the explicit transfer of information. The distinction is simply that of explicit vs implicit, or visible vs not-visible (in the model), relatedness between activities. Tight coupling, resulting from implicit relationships, is undesirable. Strong cohesion, resulting from explicit relationships, is desirable.

The UM discusses coupling and cohesion in four sections: 6.6.3 "Relationship Between Coupling and Cohesion", 6.6.4 "Metrics Based on Coupling and Cohesion" (The discussion of Coupling was included in this section.), 6.6.5 "Measures and Types of Cohesion," and 6.6.6 "Assessing Coupling and Cohesion in IDEF". The review of this material will deviate from the UM outline. We will discuss the coupling and cohesion

primitives first and then discuss the interrelationships and applicability of the concepts to the development of metrics.

2.2.1 *Coupling (UM 6.6.4.2)*

Both the UM and Dickover¹⁰ analyze coupling from two viewpoints, Nature of the Connection and Structure of the Connection. (Note: All UM material on coupling and cohesion is based on, largely verbatim, the Dickover document.) Other writings on the subject combine these viewpoints into a single set of metrics. We will discuss the nature of coupling and the structure of coupling as described in the UM and Dickover. Then, for completeness, we will discuss the combined approach, which is not based on UM material.

The distinction between nature and structure of coupling is subtle. When we are evaluating the nature of a connection we are evaluating a specific relationship between two activities and whether or not that relationship is shown explicitly by the data transfer between the activities. When we are evaluating the structure of a connection we are concerned with the explicit distribution of the information in a connection, via arrow branching, to two or more activities.

2.2.1.1 *Nature of the Connection (UM 6.6.4.2.2)*

Three categories (qualities) of coupling are defined from the viewpoint of the nature of the connection: pathological (worst), control, and normal (best). Scoring metrics are not offered. Presumably they would be: pathological = 2, control = 1, normal = 0, since a low score is the goal for coupling.

2.2.1.1.1 *Pathological Coupling*

Two functions are said to be "pathologically coupled" if one function references the contents of the other. If two activities are manipulating the same information or objects, and one of those activities requires information as to what the other is doing, then there should be an arrow between the activities explicitly providing that information. If that arrow is not present, then the activities are pathologically coupled. The example in the UM shows two activities, TAKE INVENTORY and REMOVE WAREHOUSE STOCK. The TAKE INVENTORY activity requires knowledge of what stock is being removed from the warehouse while the inventory is in progress. An arrow from REMOVE WAREHOUSE STOCK to TAKE INVENTORY is required to provide that information. Without that arrow the two activities are pathologically coupled. (The author shows a dashed arrow from TAKE INVENTORY to REMOVE WAREHOUSE STOCK (UM Fig. 6-8). We can't explain the significance of that dashed arrow.)

Pathological coupling is difficult to measure. One has to understand the relationship between two activities in order to ascertain that the relationship has not been captured by explicit data transfer in the model. This is a review-by-system-expert function and probably qualifies as a model validity measure rather than a model quality measure.

Dickover offers the following pathological coupling examples from the viewpoint of software module independence:

- A module branches into another module at a place not defined as an external entry.
- A module references non-externally declared data in another module (e.g., by address displacement).

¹⁰ Dickover, Melvin E., "Principles of Coupling and Cohesion for Use in the Practice of SofTech's Structured Analysis and Design Technique", TP039, SofTech, Inc., 460 Totten Pond Rd., Waltham, MA 02254, March 1976

- Modification of one module's code by another module.

The first item is a flow concept with no obvious counterpart in an IDEF₀ model. If we consider externally declared data to be data which shows on the parent diagram, then a tunnelled arrow would qualify as a reference to non-externally declared data. The code modification example has no obvious counterpart in an IDEF₀ model.

2.2.1.1.2 *Control Coupling*

Two functions are "control coupled" if data from one influences the flow of control of the other. This is not particularly well worded. It's perfectly OK for data from one activity to influence the flow of control of another activity. Dickover's wording is, "Two activities are control coupled if one activity passes an "element of control" datum to the other." Examples of elements of control are flags, switches, and function selection codes. The issue is whether or not the activity which is sending the data is utilizing knowledge of how the receiving activity is going to use the data. It's OK to send a message that says, "I failed". But, if the message is, "I failed, print error message", then control coupling exists. The sending activity is telling the receiving activity what to do with the information.

Control coupling can often be spotted by the presence of a verb phrase in the arrow label.

2.2.1.1.3 *Normal Coupling*

Two functions are "normally" coupled if the data connections between them exist independently of the contents of either function, or if there is no influence on the flow-of-control from one function to the other. In other words, the coupling is normal if it is neither pathological nor control. The UM wording should be, "... and if ...", rather than, "... or if ...".

2.2.1.2 *Structure of the Connection (UM 6.6.4.2.3)*

Three categories (qualities) of coupling are defined from the viewpoint of the structure of the connection: environmental (worst), record, and abstract (best). No scoring metrics are offered. Presumably they would be: environmental = 2, record = 1, abstract = 0.

2.2.1.2.1 *Environmental Coupling*

Two functions are environmentally coupled if they both reference a generic data source, without clearly specifying the relation of a particular piece of data to any box. In other words, an arrow branches into carbon copies of itself and goes to several boxes.

This is an important quality issue because it is the single most common problem encountered in IDEF₀ models. The problem is particularly serious when environmental coupling is used to provide controls to several boxes.

Environmental coupling violates the "all of the tokens" principle which states that when an arrow constrains an activity, all of the tokens of information in the arrow are involved in the constraint, not just some of the tokens. The author is using the power of ambiguity to hide the fact that the interface is only vaguely understood.

Dickover states that a degenerate form of environmental coupling occurs when a variable, or external datum, is shared between activities. In other words, an arrow splits into carbon copies of itself and goes to several boxes, each of which use all of the

tokens in the arrow. We'll see, in Section 2.2.1.3, that this is labeled "external coupling" by some authors.

2.2.1.2.2 *Record Coupling*

Two or more functions are record-coupled if they reference a non-global set of data, and do not necessarily depend on having all possible interfaces displayed, as in environmental coupling. Dickover's wording is, "... and do not necessarily need every datum in the structure", i.e. do not use "all of the tokens". He offers as examples; passing record buffer pointers, data structure names, or sharing access to a LABELED COMMON block.

The distinction between environmental and record coupling appears to simply relate to how global the set of data is. In an IDEF₀ model this would correspond to how generic, or vague, the arrow label is. This is pretty much an eye-of-the-beholder issue.

2.2.1.2.3 *Abstract Coupling*

Two elements have abstract coupling if the relation between a particular element and its potential "constituents" has been clearly specified. The introduction of "elements" and "constituents" terminology is confusing. Dickover's wording is, "Two activities have abstract coupling if all of the arrow's data tokens are necessary for a complete definition of the interface between the activities." This is the "all of the tokens" concept. Simply speaking, abstract coupling exists when the coupling structure is neither environmental nor record.

2.2.1.3 *Combined Viewpoint*

The center column of Figure 1 depicts the coupling rankings found in the literature when no distinction is made between nature and structure of the connection. The terminology is different. The correspondences are indicated in the figure. Data coupling is best, and is simply defined as not one of the other five categories. Consequently, data coupling corresponds to both normal and abstract coupling. External coupling corresponds to the degenerate case of environmental coupling in which all of the tokens are used. It is not obvious why external coupling is ranked worse than record coupling. In fact, although rankings are always indicated in the literature, the rationale for the rankings is usually missing. Intuitively, one would expect pathological, or content, coupling to be the worst. And finding environmental, or common, coupling second worst also seems logical. At least, with environmental coupling, an arrow is depicted on the diagram. With pathological coupling there usually isn't even an arrow. But, there is no obvious rationale for the relative rankings between stamp (record), control and external coupling.

<u>Nature</u>	<u>Integrated</u>	<u>Structure</u>
Normal	Data (0)	Abstract
	Stamp (1)	Record
Control	Control (2)	
	External (3)*	
	Common (4)	Environmental
Pathological	Content (5)	

*Reference the same Global Data Item.

Figure 1. Coupling Rankings

2.2.1.4 *Coupling Summary*

Pathological, or content, coupling is considered to be a model validity measure rather than a model quality measure. However, it is worth noting that some instances of tunnelled arrows are an indication of pathological coupling on the parent (diagram) of the diagram upon which the tunnelled arrow appears.

The distinction between environmental (common), external, and record (stamp) coupling is pretty much in the eye-of-the-beholder. One's attitude regarding how global the data is depends on his interpretation of how generic, or vague, the label is. We believe this is effectively captured by evaluating the extent of the distribution of the data to boxes. When an arrow splits into carbon copies of itself and goes to a couple of boxes the problem is not severe. When the arrow goes to many boxes (We've seen an arrow go to as many as sixty-five boxes in a model.) the problem is severe. When the arrow is a control the problem is worse than when it is an input or mechanism. This is not the way to provide "a control on every box". Our metric will simply score coupling structure as a function of how many boxes the arrow goes to. The more boxes an arrow goes to the worse the coupling structure. We expect this approach to have a high correlation to how global the data is. In general, the more global the data the more boxes it will go to.

Control coupling will be handled on a case by case basis. We can spot control coupling by noting verb phrases, commands, flags, switches, and the such, in the arrow labels.

2.2.2 *Cohesion (UM 6.6.5)*

Cohesion is a measure of the strength of the relationships between boxes (activities) on a diagram. The measure is based on the nature of the interconnections (arrows) between the boxes.

The UM identifies seven types of cohesion: Coincidental, Logical, Temporal, Procedural, Communicational, Sequential, and Functional --- listed here from weakest to strongest. We find it more convenient to discuss them in the reverse of that order, i.e., from strongest to weakest.

2.2.2.1 *Types of Cohesion*

2.2.2.1.1 *Functional Cohesion*

The strongest form of cohesion is functional, or control, cohesion. Functional cohesion exists when one box controls another box (Figure 2).

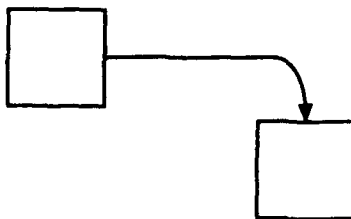


Figure 2. Functional Cohesion

2.2.2.1.2 *Sequential Cohesion*

Sequential, or input, cohesion exists when one box provides input to another box (Figure 3).

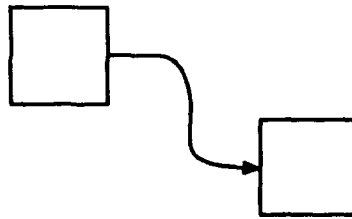


Figure 3. Sequential Cohesion

2.2.2.1.3 *Mechanism Cohesion*

Mechanism cohesion is not addressed in the UM, nor in any other writings on the subject of cohesion. Mechanism cohesion exists when one box provides a mechanism to another box (Figure 4).

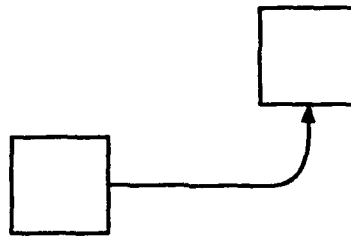


Figure 4. Mechanism Cohesion

Original writings on cohesion came from the software community. The viewpoint was software module independence. Mechanism interfaces didn't exist between software modules. There was no motivation to consider mechanism cohesion. In manufacturing technology and enterprise modeling mechanism interfaces are relatively common. So, mechanism cohesion must be considered. Since mechanism cohesion is depicted as a direct interface between two boxes it is a strong form of cohesion. Whether mechanism cohesion is weaker or stronger than sequential cohesion is debatable. We suspect that this issue may be application dependent. In some models, depending on purpose, viewpoint and context, mechanism cohesion may be as strong as functional cohesion.

Mechanisms have taken a back seat to controls and inputs throughout SADT and IDEF₀ history. We have the basic rule that every box must have a control (and the requirement to show the controls in the diagrams). It is also true that every box must have a mechanism. You can't perform the activity without it. But we are not required to show the mechanism in our model except when it suits us to do so. In reality a mechanism is not necessarily weaker than other constraints. It has merely been designated weaker, implicitly, by the rules of the methodology.

2.2.2.1.4 *Communicational Cohesion*

Communicational cohesion exists when boxes use or make the same data (Figure 5). Actually, the original writings on the subject defined communicational cohesion as occurring when two activities use the same data. The inclusion of two activities which make the same data under the heading of communicational cohesion occurs only in the IDEF₀ UM.

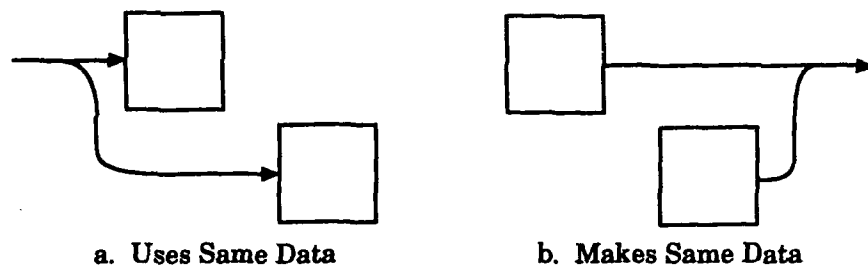


Figure 5. Communicational Cohesion

Communicational cohesion is considered the strongest form of cohesion which is not based on a direct interface between one box and another. However, there is a need for a more specific definition of what, exactly, constitutes communicational cohesion.

What is meant by "same" data? Does this mean all of the tokens of the same data? If so, then it's difficult to imagine two activities making the same data. Perhaps that's why making the same data was never included as part of communicational cohesion in previous writings. Also, using the same data would require external coupling --- one of the worst forms of coupling. We find it paradoxical that a strong form of cohesion requires a poor form of coupling for its existence.

Let's digress further and take a look at the probable motivation behind this paradox. The original writings came from the software community and the concern was module independence. When we take two strings of code which use the same data and assign them to the same module we have eliminated external coupling, between that module and other modules. That makes the module more cohesive. But when we decompose the module and view the strings of code as (child) modules we have, at that level of decomposition, the external coupling. We didn't really eliminate the external coupling. We just moved it down a layer in the hierarchy. The point is, that's a good thing to do! The severity of a coupling structure problem is a function of how deep in the hierarchy it occurs. Our coupling metric needs to take this into consideration.

Another interpretation of "same" data is bundled data. In Figure 6a bundled data, "A", arrives and splits into data, "B", to box 1, and data, "C", to box 2. Are boxes 1 and 2 using the "same" data, i.e., "A"? Is this communicational cohesion? In Figure 6b box 1 makes "D", box 2 makes "E" and then these are joined into "F". Are boxes 1 and 2 making the "same" data, i.e., "F"? Is this communicational cohesion? Our interpretation is that the author's decision to bundle data doesn't make it "same" data. Thus, the Figure 6 examples are not communicational cohesion. We believe that the author's desire to join data provides a stronger rationale for cohesion than the desire to split and use bundled data as input. That is, we don't believe "uses same data" and "makes same data" are equal strength cohesion concepts.

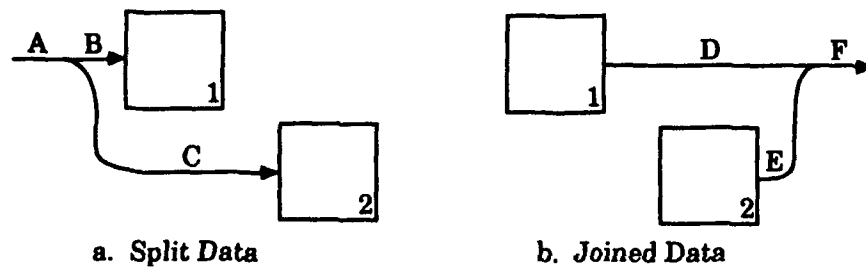


Figure 6. Interpretation of "Same" Data

2.2.2.1.5 *Procedural Cohesion*

Procedural cohesion is the grouping together of activities which are performed during the same part of a cycle or process. Dickover's wording is, "... performed during the same phase or iteration."

There is no specific construct which depicts procedural cohesion. In fact, since neither time nor sequence is implicit in an IDEF₀ diagram, there is no particular reason to expect such a construct.

The UM offers Figure 7 (UM Figure 6-14) as an example of procedural cohesion. The cohesion, between boxes 1 and 2, exists by virtue of their relationship to box 3 rather than a relationship to each other. This doesn't appear to have anything to do with "same cycle or process". The two boxes are included in the diagram because they both activate a third box. We note that A and B could also be controls, or mechanisms, or even a mix.

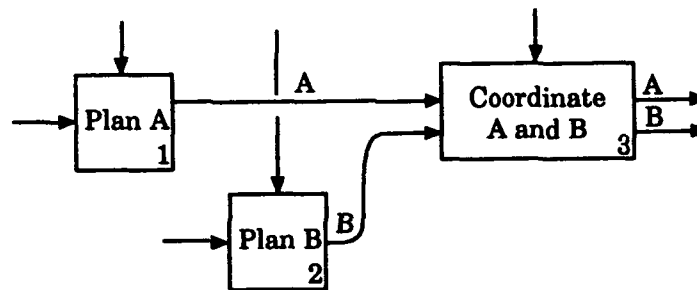


Figure 7. Procedural Cohesion Example

The real issue of significance here arises when one attempts to come up with a cohesion score for the whole diagram. Box 1 and box 3 have strong (sequential) cohesion, which justifies placing them on the same diagram. Box 2 and box 3 also have strong (sequential) cohesion, which, of course, justifies placing them on the same diagram. This causes box 1 and box 2 to, coincidentally, appear on the same diagram. That's fine. But, why identify that as yet another type of cohesion? Can cohesion between box 1 and box 2 be inherited from cohesion between box 1 and box 3 and from cohesion between box 2 and box 3? Our metric will not treat this as (additional) cohesion.

2.2.2.1.6 *Temporal Cohesion*

Temporal cohesion is the grouping together of activities which are performed at the same time --- activities which could be performed in parallel. Note that boxes 1 and 2 in Figure 7 can be performed in parallel. In fact, the original source of that figure is Dickover, and in that document it was titled, "Representation of temporally/procedurally bound diagram".

It is debatable whether the distinction between procedural and temporal cohesion is useful.

2.2.2.1.7 *Logical Cohesion*

Logical cohesion, often referred to as "decomposition by type", occurs when activities are grouped together because they are similar, or members of the same set, rather than because of any functional relationship. Decomposition by type occurs frequently in IDEF₀ models. Figure 8 is a notable example (from 9, Chapter 29. We frequently refer to the model, from which Figure 8 is a part, herein, as the "Manufacture Product" model). As in the figure, decomposition by type is frequently accompanied by

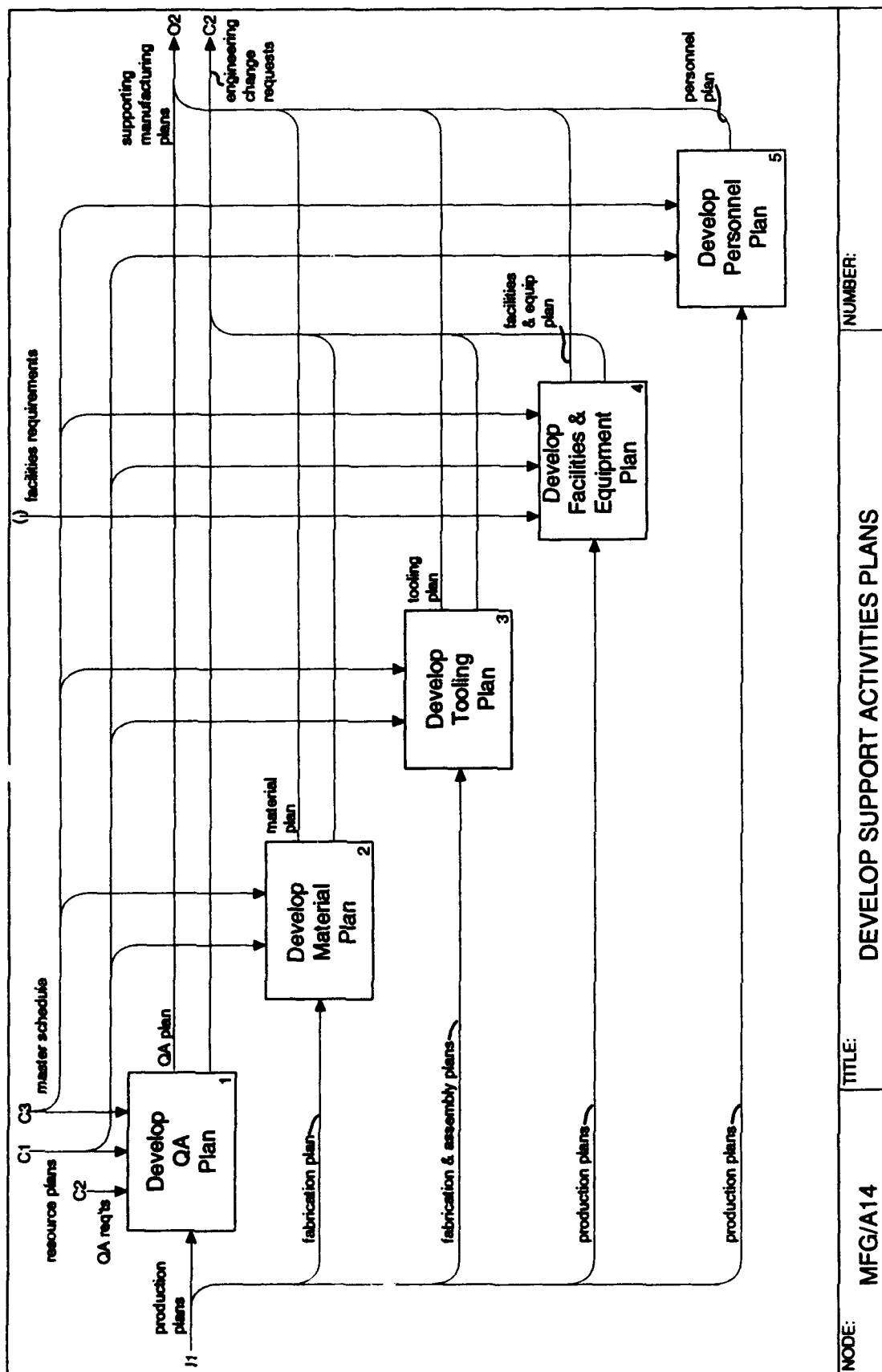


Figure 8. Logical Cohesion, "Decomposition by Type"

environmentally coupled controls and/or inputs, and the outputs are often joined into a single arrow, thus giving the illusion of cohesion.

2.2.2.1.8 *Coincidental Cohesion*

Coincidental cohesion is the name given to no cohesion at all. Any cohesion is a coincidence.

2.2.2.2 *Cohesion Metrics*

The UM and Dickover suggest the same metrics for the primitive types of cohesion: Coincidental = 0, Logical = 1, Temporal = 2, Procedural = 3, Communicational = 4, Sequential = 5, Functional = 6. If we add Mechanism = 5, then Sequential becomes 6 and Functional becomes 7.

It is stated that, "**Rating values for cohesion are additive**" (UM 6.6.6). This must be applied with the proverbial "grain of salt". Here are some examples:

We noted, in section 2.2.2.1.5, that if we apply a rating value to procedural coupling then we are giving additional "bonus points" in situations where cohesion has merely been inherited from other cohesion. In Figure 7 the sequential cohesion between box 1 and box 3 would score a "6", the sequential cohesion between box 2 and box 3 would score a "6", and the procedural cohesion (interpreted as being between box 1 and box 2) would score a "3", for a total of "15" for the diagram. We don't believe the extra 3 points were "earned". We don't believe that Figure 7 exhibits superior cohesion to Figure 9, for example, which would score "14" for two occurrences of functional cohesion.

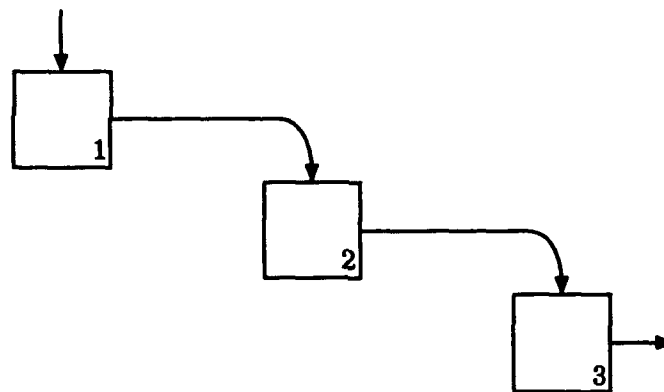
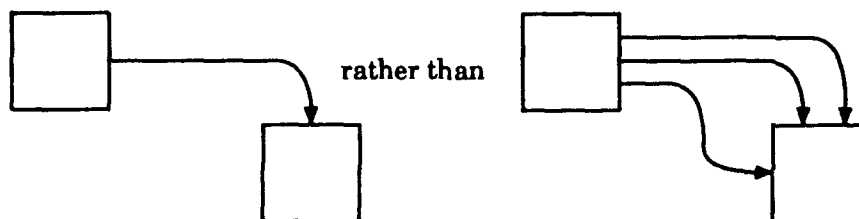


Figure 9. Additive Functional Cohesion

In UM Section 6.4.3 we find:

Bundle arrows with the same source and the same destination unless the arrow is of such importance that making it part of the pipeline would decrease clarity.



But, does the version on the right score "20", while the "preferred" version on the left scores a "7"?

Should Figures 9 and 10 both get the same score? Both have exactly two functional cohesion arrows.

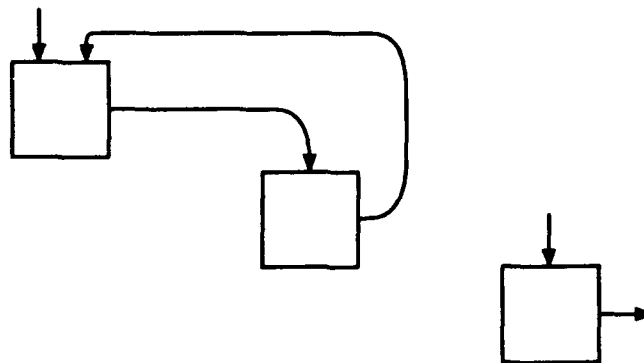


Figure 10. For Comparison with Figure 9

What should the score be when the cohesion is provided by environmental, stamp or external coupling (Figure 11)?

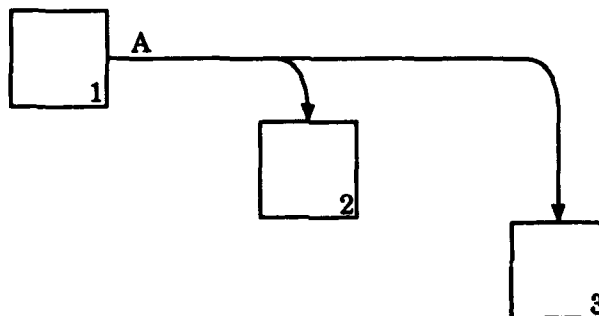


Figure 11. Functional Cohesion Provided by Environmental Coupling

Should this receive full score? Does it get the same score as Figure 9?

In situations where the cohesion is not provided by direct interfaces, i.e., procedural, temporal and logical, do we interpret the primitive cohesion metric as applicable to pairs of boxes or to the whole diagram? In other words, when cohesion is provided by direct interfaces it is easy to see what we're "adding up" to get a whole diagram score. But when the cohesion is not provided by direct interfaces, what are we "adding" to what? Does a three-box diagram with logical cohesion get the same score as a five-box diagram with logical cohesion or should the scores be different?

Note that, in general, if we add cohesion primitives to get a diagram score then diagrams with several boxes will have higher cohesion scores than diagrams with a few boxes. We don't believe the cohesion concept has anything to do with the number of boxes on a diagram. The cohesion concept evaluates the relatedness of the boxes.

Without belaboring the point, it is clear that simply adding primitives to get a whole diagram score isn't valid. If a fine-grained metric is desired for rating the cohesion of whole diagrams, the algorithm which generates that metric will be more complex than the simple addition of primitives.

Another related concern is the relative values of the primitives. All of the cohesion primitives involving a direct interface between two boxes, i.e., functional, sequential and mechanism, are "good" cohesion. How important is it to rank one type of

direct interface higher than another? For example, in UM Section 6.2.2, we have, **"If it is uncertain whether an arrow is a control or an input, make it a control."** In those cases in which the arrow should be an input, do we give the author an extra rating point for being "uncertain" and making it a control? Or, for that matter, should we encourage authors to make everything controls for higher quality scores? Obviously not. We commented, in Section 2.2.2.1.3, that in some scenarios mechanism cohesion may be as strong as sequential, or even functional cohesion.

At the other end of the spectrum is the question of the utility of making a scoring distinction between hypothetical levels of incohesiveness, e.g., procedural, temporal logical. All are reasons, or excuses, for putting boxes, which don't talk to each other, together on a diagram. It is debatable whether any of these warrant higher cohesion ratings than the others. It's kind of like trying to define pornography --- the pictures are the same, the difference is the author's intent.

2.2.2.3 *Cohesion Summary*

Cohesion is a valid quality consideration for IDEF₀ diagrams. Cohesion provides a potential metric for a substantive quality factor, decomposition quality. The baseline definition of cohesion primitives by Dickover and others was a valuable step in the right direction. We believe the attempt to define seven levels of granularity is overkill --- two or three levels is probably sufficient. Simple addition of primitives to obtain a whole diagram score is not valid. A more complicated procedure is required.

2.2.3 *Coupling, Cohesion, and Complexity*

In UM Section 6.6.3 we have,

"When components of a diagram or model are highly coupled to each other, the model is more complex. When components are composed of unrelated (incohesive) elements, the model is more complex."

The New College Edition of the American Heritage Dictionary defines complex as "Consisting of interconnected or interwoven parts", "Involved or intricate in structure". Synonyms include, "complicated, intricate, involved, tangled, knotty". The dictionary goes on to make a subtle usage distinction between complex and complicated: "Complex often implies many varying parts; complicated stresses elaborate relationship of parts rather than number."

Let's take another look at Figure 8. The components are highly coupled; the controls are environmentally coupled and the inputs are a mix of environmental and record coupling. The components are composed of unrelated elements; logical cohesion, members of the same set --- "plans". Some readers will consider Figure 8 to reflect communicational cohesion --- uses/makes same (bundled) data. But is the diagram complex? No! The diagram has a lot of problems. But complexity isn't one of them. The diagram doesn't have interconnected parts nor exhibit an intricate structure. The diagram is simple to understand because it doesn't say very much.

So, what's the point? Are we suggesting that poor cohesion and coupling have nothing to do with complexity? No. The issue is absolute complexity vs relative complexity. We believe absolute complexity is best measured in terms of activations per box (see Section 3.2.3). Relative complexity, on the other hand, is a measure of the balance between complexity and information content on a diagram. We submit that Figure 8 is a very complex diagram, relative to its information content. And that balance, or unbalance, between complexity and information content is what the coupling and cohesion metrics measure. We consider coupling and cohesion to be measures of amplification. The "just right" amount of amplification is a trade-off between information content and complexity.

2.2.4 *Coupling and Cohesion Summary*

The concepts of coupling and cohesion are sufficiently related that considering them separately may not be appropriate. We noted in Section 2.2.2.1.4 that communicational cohesion (considered to be good cohesion) required external coupling (considered to be poor coupling) for its presence on a diagram. We noted, in Figure 11, another example of "good" cohesion being provided by "bad" coupling.

We can either redefine "good" and "bad" in terms of combined cohesion and coupling metrics, or we can score them separately and account for conflicts and tradeoffs in some other manner. We prefer the latter, separate scoring, because it earmarks the source of a problem as being a cohesion or coupling problem. This should be of greater value to an author in his quest to make repairs. And, this approach allows us to build on the valuable work already done on coupling and cohesion, rather than set about inventing a new wheel.

3.0 PRIMITIVE METRICS

Primitive metrics is the nomenclature we have chosen for the set of quality indicators that we know how to measure in a quantitative manner. The existence of a large quantity of primitive metrics, covering the spectrum of quality factors we would like to measure, establishes feasibility of the quality metrics concept. The existence of a large quantity of machine detectable primitive metrics establishes feasibility of an expert system implementation.

Metrics are based on rules. Rules are based on rationale. Some rules are absolute --- if you obey the rule the model is correct, if you disobey the rule the model is wrong. Most rules are good practice rules --- one way of doing something is better than another way of doing the same thing, but neither way is wrong. Some rules are Guru rules --- someone said something was right or wrong, or good or bad, but offered no rationale.

In this section of the Phase I Final Report we present the results of our research into primitive metrics. The Baseline material from the UM addressed primitive metrics exclusively. We have reorganized that material and enhanced it in accordance with our detailed review and evaluation comments as presented in Section 2.0. We further elaborate on that material by discussing required formalizations, machine detectability, and our preliminary thoughts on the integration of the primitives into whole diagram and whole model metrics. We comment on the distinction between real and Guru rules and on the distinction between absolute and good practice rules.

The presentation is divided into two general categories of primitive metrics: Syntax and Semantics; and Decomposition Quality Factors. We believe that the Decomposition Quality Factors are a genuinely distinct category of quality issues which do not fall under the headings of syntax or semantics.

3.1 *Syntax and Semantics*

We find it inconvenient to discuss syntax and semantics separately, i.e., a section on syntax followed by a section on semantics. Why would we want to talk about, "an arrow entering the top of a box", when we can introduce some semantics first and talk about, "a control arrow"?


Our presentation format will be to divide the discussion into topics. Then, within the discussion of each topic we will distinguish between syntactic and semantic measures. The distinction is not always useful. We comment on the utility of the distinction, from a metrics perspective, on a case-by-case basis. We found that the syntax and semantics primitives are conveniently categorized into the four topics: Diagrams, Boxes and Arrows, Labels and Titles, Codes and Numbers.

3.1.1 *Diagrams*

An IDEF₀ model is a set of related diagrams. In particular, an IDEF₀ model is rendered on a set of forms. The set of forms includes a Cover Sheet, a Node Index/Contents Sheet, a Standard Diagram Form, and others. We are going to limit our discussion to the Standard Diagram Form.

The Standard Diagram Form was omitted from the UM. It was supposed to be UM Figure 5-5. Figure 12 is a reproduction of the original SADT Diagram Form. It is identical to the form used for IDEF₀ models.

The Standard Diagram Form consists of three fields, as annotated in Figure 12: a Working Information field at the top of the form, a Message field in the center of the

USED AT: AUTHOR: PROJECT: NOTES: 1 2 3 4 5 6 7 8 9 10	DATE:	WORKING	READER	DATE	CONTEXT:				
	REV:	DRAFT							
		RECOMMENDED							
		PUBLICATION							
<div style="text-align: center;">  </div>									
NODE:		TITLE:		NUMBER:					

Working
Information
Field

Message
Field

Identification
Fields

Figure 12. SADT Diagram Form

form, and Identification fields at the bottom of the form. A delivered model typically deletes the Working Information field and leaves the NUMBER (sometimes called "C-NUMBER") box of the Identification field blank (as in Figure 8, for example). We will limit our discussion to the Message field and the NODE and TITLE boxes in the Identification field.

The Message field may contain an IDEF₀ diagram, a glossary, text, or "for exposition only" (FEO) material. There are no formal IDEF₀ rules for glossaries, text nor FEOs (except that text is limited to one page). We will limit our discussion to IDEF₀ diagrams.

So, when we say, "An IDEF₀ model is a set of related diagrams", we are referring, solely, to Standard Diagram Forms upon which the Message field contains an IDEF₀ diagram. It is important to make this distinction because the IDEF community freely uses the term "diagram" when referring to Standard Diagram forms, regardless of the content of the Message field. We use the term "diagram" to refer only to forms upon which the Message field contains an IDEF₀ diagram.

Metrics will be developed for diagrams, including the NODE and TITLE fields, but excluding the Working Information fields and NUMBER field. Metrics will not be developed for FEOs, glossaries nor text except that the NODE and TITLE fields on FEOs, glossary and text will be checked for consistency with the NODE and TITLE field of the corresponding diagram.

There are two kinds of diagrams: Context Diagrams and Decomposition Diagrams. The latter terminology is our own invention for "diagrams other than context diagrams". There is no universally accepted terminology for this set. We want to follow our own advice by describing context diagrams separately, so as to avoid the necessity of frequent statements, "Except for the A-0 diagram ...", throughout the metrics presentation.

3.1.1.1 Context Diagrams

The presentation of a finished IDEF₀ model is a set of diagrams organized in the form of a hierarchical decomposition. Context diagrams are diagrams which are thought of as being "above the top" of the hierarchical decomposition. One context diagram is required for every IDEF₀ model. The required context diagram is called the "A-minus-zero" (A-0) diagram. Additional context diagrams; "A-minus-one" (A-1), "A-minus-two" (A-2), etc., are allowed and occasionally included in a model, but not required.

3.1.1.1.1 A-0 Diagram

Every IDEF₀ model must have an A-0 diagram. The purpose of the A-0 diagram is to bound the model. The A-0 diagram distinguishes that which is part of the model (inside the model) from that which is not part of the model (outside the model).

The A-0 diagram depicts exactly one box. The box contains a verb phrase descriptive of the whole model function (activity). The number of the box is zero, "0". This number needn't be shown in the box. The number is implicit, whether shown or not. The arrows shown on the diagram depict the interfaces between activities in the model and activities which are outside the model. Like all boxes on all diagrams, the A-0 box must have at least one control arrow and at least one output arrow. The arrows on an A-0 diagram are not required to have ICOM codes, even when there are higher level context diagrams.

The A-0 diagram is the only diagram in an IDEF₀ model which has a specific requirement for textual material. The A-0 diagram must contain a statement of the Purpose of the model and a statement of the Viewpoint of the model.

Metrics applicable to the A-0 diagram may appear trivial, but they are important. The most important metric is the existence of the diagram in the first place. The failure to bound a model, to place it in the context of a larger frame of reference, is usually terminal. Equally important, in spite of apparent triviality, is the presence of arrows on the diagram (Yes, we have seen A-0 diagrams with no arrows.). The box and the arrows must, of course, be labeled.

Metrics for Purpose and Viewpoint are, for the most part, human review items. The presence of the words "Purpose" and "Viewpoint", followed by character strings, presumably text, is machine detectable. But, the substantive quality issue is what the words say. Some guidance is available for review of the purpose and viewpoint. We recommend the discussion in Section 1 of Marca⁹. The purpose of a model is to answer questions about the system being modeled. Although an over-simplification, the purpose should disclose the questions and the viewpoint should disclose who is answering the questions. The viewpoint is that of the system expert upon whose expertise the model is based. The purpose and viewpoint, together, should suggest a decomposition strategy and should suggest the appropriate "natural language" for labels. It is our experience that the concise purpose and viewpoint statements on most models fall short of this information content. There is no requirement for conciseness of these items.

3.1.1.1.2 Higher Level Context Diagrams

Higher level context diagrams differ from Decomposition Diagrams in only two respects. Data structure connectivity (ICOM coding) is generally omitted. And, only one box on each diagram is decomposed. Specific rules for these diagrams are not included in the UM. We consider the Decomposition Diagram syntax and semantics rules for boxes, arrows and labels to be applicable. The coupling, cohesion and complexity quality factors are applicable.

3.1.1.2 Decomposition Diagrams

All of the material in the remainder of the Primitive Metrics section of this report exclusively addresses Decomposition Diagrams.

3.1.2 Boxes and Arrows

IDEF₀ diagrams are comprised of boxes and arrows. It is somewhat difficult to discuss boxes and arrows separately. It is particularly difficult to discuss boxes independently from arrows. Doug Ross, the creator of SADT, has said, "A box is defined completely and unambiguously by its arrows." "It is entirely the structure of arrows that determine what the box is; that determination is entirely controlled by context."¹¹ Thus, in the following discussion we will not have a great deal to say about boxes. We will have a lot to say about arrows.

3.1.2.1 Boxes

3.1.2.1.1 Box Syntax

There are no specific syntax for boxes. A box is a box is a box As we noted previously, there would be no model --- no diagrams --- without boxes. But, there are no specific rules for required boxes. If one wanted to stretch a point, for the sake of formality, he could argue that the "three to six boxes per diagram" rule is a required boxes rule. In fact, you could argue that "at least three" is completeness and "no more than six" is correctness. We consider the "three to six boxes per diagram" rule to be an amplification

¹¹ Douglas Ross Talks About Structured Analysis", (Interview), Computer Magazine, July 1985, pp. 80-88

rule. The rule is global (whole diagram) syntax. But, the underlying rationale behind the rule is an amplification concept.

3.1.2.1.2 *Box Semantics*

The key semantics for boxes is the meanings of the sides of a box, Figure 13. We believe some formalism is required relative to the distinction between inputs, controls, and mechanisms.

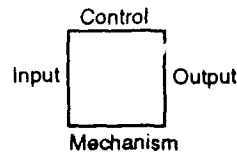


Figure 13. Meanings of the Sides of a Box

We find the "consumed or manipulated" characterization for inputs to be unwieldy. It is quite common for a control to be manipulated to produce an output. The issue --- based on common practice rather than stated rules --- appears to be destructive vs non-destructive utilization. This causes almost everything to be characterized as a control, because we tend to save everything whether we need to or not. The purpose of inputs and outputs in an IDEF₀ diagram is to specify a transformation, a before and after state of something. We don't believe that the continued existence of the before state, for other purposes or uses, after the transformation accomplished by some specific activity has occurred, renders the before state a control. The before state is an input, to this activity, which happens to have other uses, and is, therefore, saved.

There are also formalism problems relative to the distinction between controls and mechanisms. A control is a constraint which is not transformed by the activity. So is a mechanism! Is the distinction merely intuitive? As we've mentioned previously, the original writings from the software community pretty much ignored mechanism theory. Controls were always data. Mechanisms were never data. So, conflicts between the definition of mechanisms and controls didn't occur. But, in the manufacturing technology and enterprise modeling domains, in which arrows can be objects as well as data, or information, ("artifacts" is also gaining vogue) the distinction between controls and mechanisms becomes fuzzy. We believe the "controls are always data" and "mechanisms are never data" distinction is a possibility worth considering. There are counter-examples. If mechanisms can be "tools" (a popular utilization), then how do we handle forms and templates? Aren't those "data"? How about checklists ... mechanisms or controls? How about "sunshine" in the classical "Grow Vegetables" model? That's not data. Is it a control, or a mechanism?

We're not going to resolve these issues in this quality metrics report. It is well beyond the scope of our research. But, one of the advertised results of the quality metrics research is recommended formalizations. We believe there is some useful research to be accomplished in the formal definitions of inputs, controls, and mechanisms.

Another semantics issue relative to boxes is box layout on a diagram (global, or whole diagram, semantics). The recommended layout of boxes on a diagram is diagonal, from top-left to bottom-right. This serves two purposes. It is the layout which best caters to the minimization of clutter --- arrow interfaces, arrow crossings, etc. The diagonal layout is also advertised as representing precedence relationships. A box which is placed above and to the left of another box is presumed to be a stronger constraint (relative to the rest of the diagram) than a box which is located lower and more to the right. This is a good practice rule, rather than an absolute rule. There is no hard and fast requirement for

diagonal layout of boxes. The two purposes, clutter reduction and precedence relationships, can be in conflict. We believe the precedence relationship purpose is the more important consideration. We frequently refer to this purpose as the "subliminal pitch". Readers tend to assume the precedence relationship whether, or not, it was intended. Good modelers exploit this. Thus, it becomes a model quality issue.

It is worth noting that the diagonal layout does not cater to mechanism interfaces between boxes. A mechanism interface is more pleasing-to-the-eye if the source is below and to the left of the sink. Again, we encounter the problem of original writings ignoring mechanisms. We need to treat left-right precedence as stronger than upper-lower precedence in order to cater to mechanism interfaces.

In summary, metrics relative to boxes on IDEF₀ diagrams include the "three-to-six boxes per diagram" rule; the distinction between controls, inputs and mechanisms; and the detection of precedence relationship inconsistencies. The three-to-six box rule is easily detected, in fact machine detectable. Whether the rule is treated as global syntax or amplification is immaterial in the primitive metrics discussion. We will raise the issue again in the discussion of whole diagram and whole model metrics. The distinction between correctness of controls, inputs and mechanisms is not machine detectable. This is an expert review item. Precedence relationships are frequently machine detectable. The inconsistencies occur as conflicts between precedence and cohesion relationships. An example is a diagram which portrays a feedback path between two boxes when there was no forward path between the boxes in the first place. A more subtle example is shown in Figure 14. The layout suggests that box 1 is stronger than box 2. But, the cohesion metrics indicate that box 2 is stronger than box 1, since control (functional) cohesion is stronger than input (sequential) cohesion.

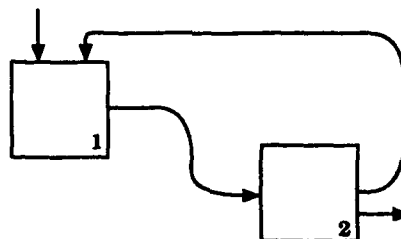


Figure 14. Precedence - Cohesion Inconsistency

3.1.2.2 Arrows

3.1.2.2.1 Arrow Syntax

Arrow syntax includes rules for required arrows, arrow construction rules, rules for bundling, branching and joining arrows, and rules for tunnelled arrows.

3.1.2.2.1.1 Required Arrows

Required arrows include a control on every box and an output from every box. Every box must have at least one control arrow. Every box must have at least one output arrow. As we have mentioned previously, an activity cannot be performed without a mechanism. The IDEF₀ methodology does not require a mechanism arrow on every box. Showing mechanisms is strictly author's prerogative. In the software development environment that was OK. In the manufacturing technology and enterprise modeling environments it may not be OK. This is a formalism issue, beyond the scope of the metrics research reported herein. We consider this to be a significant formalism issue which needs to be addressed. There are no rules for the utilization of mechanisms in IDEF₀ models other than as an option.

Required arrows also include arrows inherited from the parent diagram. Every arrow touching a box on the parent diagram must be present on the child diagram of that box. (Tunnelled arrows are an exception, discussed later.)

Testing for the presence of required arrows on a diagram is straightforward and machine implementable.

3.1.2.2.1.2 Arrow Construction

Our recommended arrow construction rules were introduced in Section 2.1.1.2.1. We found some problems with the arrow segment definitions as presented in the UM.

There are two sources of arrows in an IDEF₀ model. An arrow can have its source inside the model as the output of a box. Or, an arrow can have its source outside of the model. Arrows arriving from outside of the model appear as inputs, controls or mechanisms on the A-O Context Diagram or can arrive elsewhere in the model as tunnelled inputs, controls or mechanisms.

Similarly there are two sinks for arrows in an IDEF₀ model. An arrow can have its sink inside the model as an input, control or mechanism on a box. An arrow sink may be outside of the model, departing the model as an output on the A-O Context Diagram or departing elsewhere in the model as a tunnelled output.

We use the terminology "direct interface" when an arrow goes from the output of one box to the input, control or mechanism of another box on the same diagram. Also, rarely, another form of direct interface, called cyclic processing (Figure 15), occurs when an arrow source and sink is the same box. Presumably, the cyclic processing sink could also be the mechanism side of the box. We've never seen that in a model.

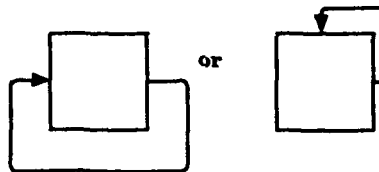


Figure 15. Cyclic Processing

Arrows which are not direct interfaces are called boundary arrows. Boundary arrows arrive at the edge of a diagram as inputs, controls or mechanisms and terminate at a box input, control or mechanism sink. Note that the role of the arriving arrow, i.e., its characterization as an input, control or mechanism, at the edge of a diagram, need not be the same as its role at the box sink. However, this role change is somewhat pathological when it occurs ... the roles are usually the same. Boundary arrows departing the diagram as outputs always originate from an output source of a box. Boundary arrows may be interfaces with other diagrams or tunnelled arrows.

We have used the terminology "arrive" at the edge of a diagram and "depart" from the edge of a diagram for boundary arrows because we don't believe the edges of a diagram should be considered to be sources nor sinks. Arrows do not originate nor terminate at the edges of a diagram. They are coming from or going to the outside of the diagram. This is even the case on the A-O diagram. We consider this to a subtle, but important, mind-set issue. This was one of the difficulties we had with the UM arrow segment definitions.

Measurement of correct arrow construction is accomplished by arrow tracing. Arrows are traced from source to sink, or from sink to source, to determine if the sources and sinks are syntactically correct. This procedure is machine implementable.

3.1.2.2.1.3 *Arrow Bundling, Branching and Joining*

From a syntax perspective we simply note that joining arrows into bundles and branching arrows out of bundles is allowed, in fact, frequently recommended. Most of the problems encountered with arrow joining and branching are either semantic or labeling problems. The arrow tracing procedures are complicated by bundling but, nevertheless, implementable.

3.1.2.2.1.4 *Arrow Tunnelling*

An arrow enclosed in parentheses either at the edge of a diagram or where it touches a box is referred to as a tunnelled arrow. It is inconvenient to discuss tunnelled arrow syntax without introducing tunneled arrow semantics.

An arrow enclosed in parenthesis where it touches a box does not appear on the child diagram (i.e., the decomposition) of the box. An arrow enclosed in parentheses at the edge of a diagram does not appear on the parent diagram which spawned the decomposition upon which the parenthesized arrow appears.

Thus, the tunnelled arrow syntax measure is a check, on a specific diagram other than the diagram containing the tunnelled arrow, to confirm that the arrow is not present.

3.1.2.2.2 *Arrow Semantics*

We noted in our discussion of boxes that there wasn't much to say about syntax. Most of the discussion was semantics. Conversely, we had a lot to say about arrow syntax and don't have much to say about arrow semantics.

An arrow depicts a constraint applied to one activity by another activity. The constraint may be exhibited as information, or data, or as the provision of material things. The arrow simply indicates which activities are constraining which other activities, by linking the activities together. It is interesting to note that the arrowheads contribute no useful information to a diagram. The direction of constraint transfer is unambiguously defined by the sides of the boxes. We know which end of the "arrow" is the source and we know which end of the "arrow" is the sink by virtue of the sides of the boxes the "arrow" touches. The arrowheads are strictly a human factors consideration. The diagrams are a lot easier to read with arrowheads.

We are aware of no semantic issues relative to required arrows or arrow construction. We have some comments on bundling and tunnelling.

3.1.2.2.2.1 *Bundling, Branching and Joining Semantics*

Bundled arrows are simply collections of arrows which are depicted in the model as a single arrow, i.e., a collection of arrows is "bundled" together. Bundles can arrive from outside the model, or be created inside the model by joining arrows together. Bundled arrows can depart from the model as outputs. Ideally, bundled arrows are decomposed into their component parts inside the model, i.e., via arrow branching, or splitting.

It is worth noting that an arrow branch does not necessarily imply the existence of a bundled arrow prior to the branch. Arrows are permitted to branch into carbon copies of themselves. This use of a branch simply depicts the same data going to more than one activity. In fact, there is a rule (UM 6.4.2) stating that arrows depicting the same data going to two (or more) activities should utilize a branch construct rather than merely having the same label (Figure 16).

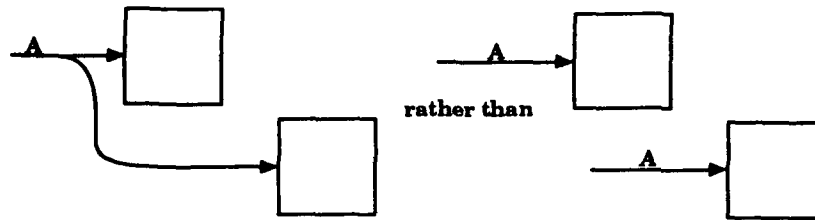


Figure 16. Connection of Open-Ended Boundary Arrows

Arrows may be bundled for a number of different reasons. We consider this to be a problem worthy of some formalization research.

One reason for bundling arrows is simply clutter reduction. Arrows are bundled to reduce the number of arrows on a diagram. The following two rules from UM 6.4.3 are examples (Figure 17). Note that there is no explicit statement that the data items on the arrows be related.

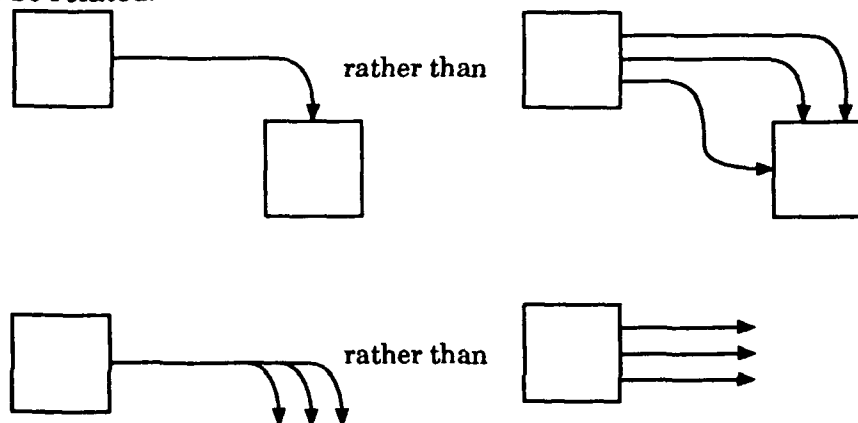


Figure 17. Bundling for Clutter Reduction

The best reason for bundling arrows is to depict data relationships. Arrows are bundled because they can be thought of as members of a set (the bundled set).

A third reason for joining arrows is to depict activation logic for an activity (Figure 18). This is based on the "all of the tokens" concept. In "a." the data A or B or both may activate box 3. In "b." the data C, which is comprised of all of the tokens of A and B, activates box 3, i.e., A and B (logical "and") are required to activate box 3. A and B may be, otherwise, unrelated.

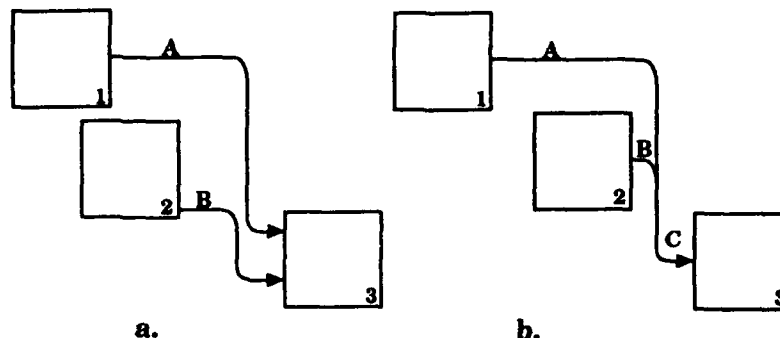


Figure 18. Bundling to Depict Activation Logic

Our concern with having more than one rationale for bundling is that it is not always apparent from context which rationale was used.

We require a consistency rule for bundled arrows which are both created by joins and decomposed by branches inside a model. Figure 19 depicts the problem. The arrows which are joined to create a bundle and the arrows which are branched out of the bundle must be the same arrows. The Manufacturing Plan arrow in the Manufacture Product model has this problem. A similar problem occurs when you can't tell whether the inconsistency exists --- a bundle is created by joins then later branches into carbon copies of itself, rather than being decomposed, and goes to several boxes.

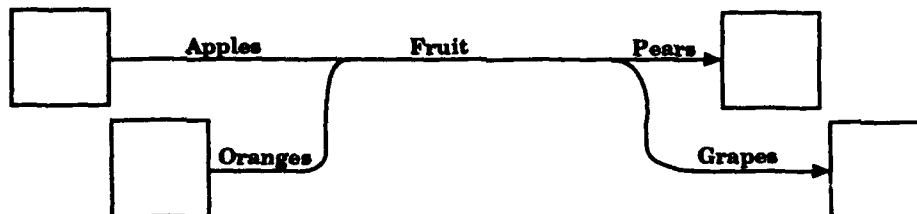


Figure 19. Inconsistent Joins and Branches

Clutter reduction rules (Figure 17), required joins (Figure 16) and consistency checks (Figure 19) are machine implementable. Legitimacy of branches, joins and bundles is a human review item, based on labeling, unsupported by formal rules stating what is, and what is not, OK.

3.1.2.2.2 *Tunnelled Arrow Semantics*

Tunnelled arrows can be used for several different purposes. There are no published rules for the correct application of tunnelled arrows. Formalizations are needed.

The term "tunnelled arrow" is applied to two unrelated concepts. The only relationship is that both are depicted syntactically by parentheses. Arrows which are parenthesized at the edge of a diagram arrive from, or depart to, the outside of the model without having appeared on higher level diagrams. This fits one's intuitive concept of arriving or departing "through a tunnel". Arrows which are parenthesized where they touch a box are simply excluded from further consideration in the model. That's not a tunnelling concept. That's more like a "black hole" concept.

The use of parentheses fits the tunnelling concept well, but is simply a handy notation for the black hole concept. In text, a parenthesized remark is something not important enough to feature in the main discussion but considered worth mentioning "in passing". That's what happens with a tunnelled arrow --- it wasn't important enough to bring down through the hierarchical decomposition but is worth mentioning on a particular diagram. One could make a strong case for parenthesizing both ends of an arrow to indicate that this is the only diagram upon which it appears. The other application is just the opposite. The arrow has been featured until now, but we have no interest in discussing it further. That's not a parenthetical remark concept.

Since there are no rules for the application of tunnelled arrows, there is no basis for metrics. We offer some opinions.

- Inputs should not be tunnelled into a model.
- Controls may be tunnelled into a model, but cannot be utilized as the sole control on a box.
- Byproducts of an activity may be tunnelled out of a model, but cannot be the sole output of the activity.

- Mechanisms may be tunnelled into a model but should not be the sole mechanism on a box (even though other mechanisms may be implicit rather than explicit due to current methodology treatment of mechanisms).
- The black hole concept should not be used except when the arrow goes to all lower level boxes. The motivation is clutter reduction.
- Tunnelling should never be used to transfer data between two boxes in the same model.

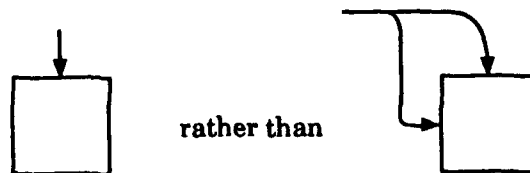
3.1.2.2.3 *Rules for Drawing Arrows*

In section 2.1.1.2.2 we stated that several rules from UM 6.3, 6.4, and 6.6.2 were applicable to the model quality discussion. We have already introduced a few where they occurred naturally in the discussion. For completeness, we list the remainder of those applicable to arrow placement and layout here, in most cases without comment.

- On any one side of a box there should never be more than four arrows.

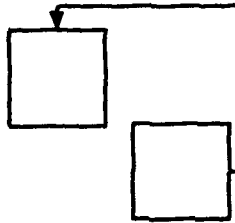
Why four? In the case of the three-to-six-boxes-per-diagram rule there was a basis in human factors writings for the rule. But what is the rationale for four arrows per side of a box? We don't have any particular objection to the rule. But it appears to be a Guru rule. Someone said the number should be "four" and that's what it has been ever since. It's traceable back to very early SADT writings, but never accompanied by rationale. Early SADT writings also say "No more than six arrows per box." Of course, back in those days you rarely had mechanisms. This rule at least had some rationale. It came from the application of the three-to-six-boxes-per-diagram rule applied to SADT data diagrams (Datagrams). Duality translated this to three-to-six-arrows-per-box on an activity diagram. We note that the rules are compatible. You can impose both "no more than four arrows per side" and "no more than six arrows per box" as diagramming rules. Some formalization, with rationale, would be nice. Our metric currently invokes the rule, "no more than four arrows per side", in accordance with the UM.

- Don't split an arrow into both a control and input to the same box.

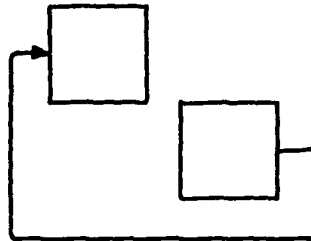


- Draw arrows along horizontal and vertical lines, not diagonally or as curves.
- Place arrow corners, intersections, and labels a reasonable distance away from boxes.
- Place extra arrowheads along arrows where needed for clarity (i.e., the syntax allows extra arrowheads).
- Don't draw arrows all the way to the margins of the diagram.

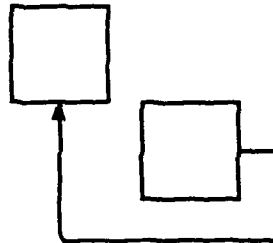
- Space parallel arrows as far apart as possible
- Control feedback should be shown up and over.



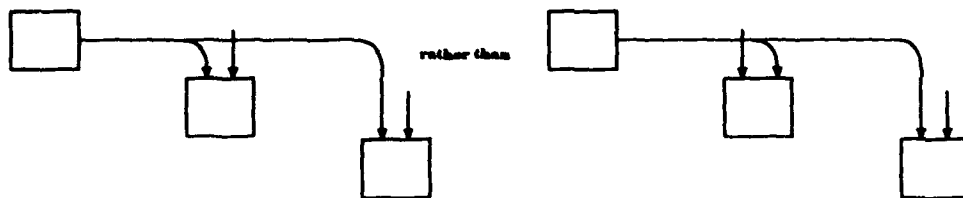
- Input feedback should be shown down and under.



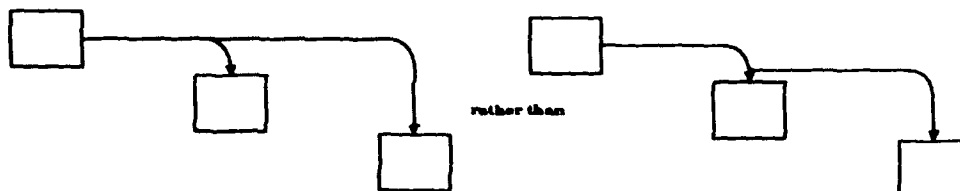
- (Addition) There is no convention for mechanism feedback. It seems intuitively apparent that the only reasonable choice is down and under.



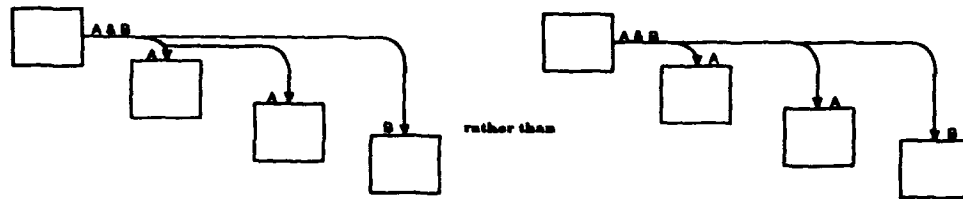
- Lay out arrows so as to minimize arrow crossings.
- If an arrow branches and feeds into several boxes, draw it at the same relative position on each box, if possible. (If possible? It's always possible. But you should not introduce an unnecessary arrow crossing to do it.)



- Minimize corners.

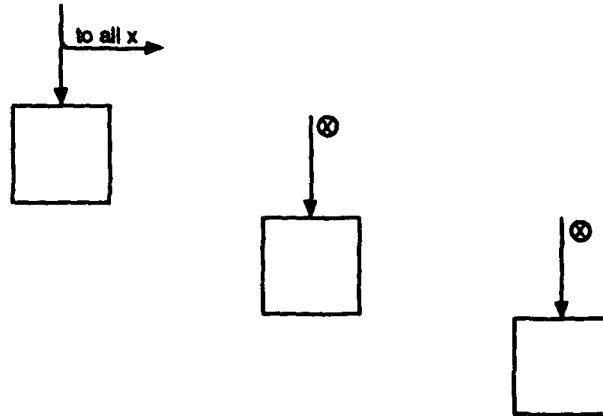


- Use expressive potential of branching arrows.

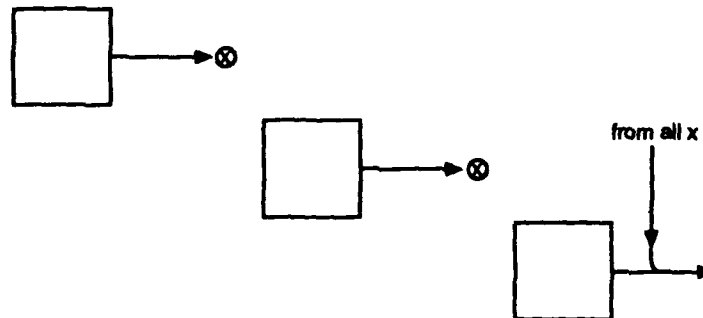


(Note discussion in Section 3.2.1.3.2)

- When an internal arrow applies identically to every box on a diagram use the "to all" convention.



- When an external arrow (output) is obtained identically from every box on a diagram use the "from all" convention.



We note that most of these rules are motivated by clutter reduction. All are machine detectable. Many are machine correctable. The greatest challenge is combining the primitive measures into overall diagram measures.

3.1.3 Labels and Titles

Boxes and arrows have labels. Diagrams have titles. In this section of the primitive metrics discussion we address the various rules for labeling and titling.

3.1.3.1 Box Labels

Syntactically speaking, boxes are labeled with verb phrases. The verb phrase is located inside the box.

There are a few good practice rules for box labels. One should avoid the use of IDEF0 keywords; input, control, output, mechanism, function, activity. One should avoid verb phrases that don't contribute information. For example, if the input is "x" and the output is "y", the verb phrase should be something more profound than, "transform x into

y". In this vein, SADT writings and the IDEF₀ UM suggest avoiding phrases like "make x". We find this rule (good practice) frequently difficult to live with. Simply substituting words such as "prepare", "develop", "manufacture", or "construct" for the word "make" hasn't really avoided the problem. We see these words a lot in IDEF₀ diagrams.

In formal writings, particularly in documents such as specifications and standards, the object (often the output) frequently has a verb form. "Specify Requirements" is preferable to "Develop Requirements Specification". "Plan Tests" is preferable to "Develop Test Plan". The verb form usually produces a more concise phrase, and avoids the "make x" form. But, often there is no choice available. What do you do with "Develop Test Procedures"? "Procedurize Test"?

Probably the most important issue relative to the labeling of boxes is that the label isn't very important in the first place. Blasphemy? Doug Ross says, "The boxes, which are supposedly the focal point of the whole diagrammatic exercise, don't need names at all. The reason is that every box is going to be detailed further on the next lower diagram. What it means is what is shown there. Only the lowest level of boxes, the atomic level, needs names. At the higher levels the names are a convenience and an aide memoir, but are logically redundant." (Also from 11.)

A few other rules are basically common sense. One should not use the same verb phrase for two different boxes. The most common occurrence of this problem is when a box on a child diagram has the same label as the parent diagram. This occurs due to the common practice of labeling a process the same as the last step in the process. "Drive car" for example might have activities such as "insert key in ignition", "turn key", "press accelerator", "put car in gear", ... then, last but not least, "drive car". Sometimes boxes have identical labels in disguise. The Manufacture Product model, for example, has boxes labeled "Plan Production" and "Develop Production Plans".

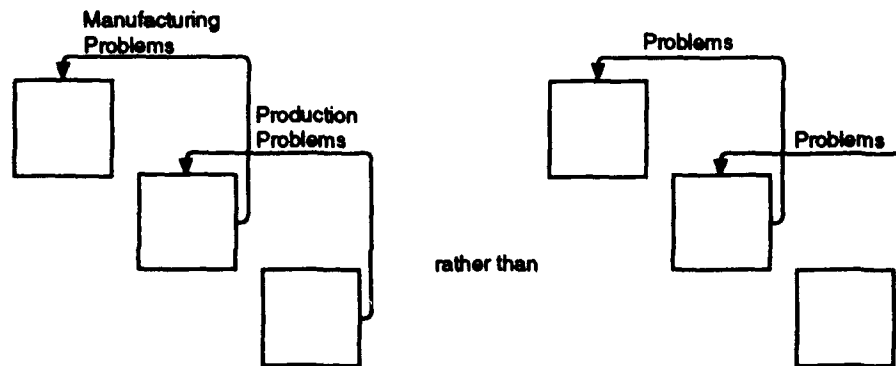
3.1.3.2 *Arrow Labels*

Arrows are labeled with nouns or noun phrases. The association of a label with an arrow is depicted by the proximity of the label to the arrow or by the use of a "squiggle" (yes, squiggle is in the dictionary) between the label and the arrow.

Again we have the avoidance of key words; control, input, output, mechanism, data, information. We have the desire for conciseness and recommend the use of a Glossary to achieve label conciseness on the diagrams.

We have two very important rules for arrow labels (UM 6.6.2):

- The label of an arrow on the parent diagram and the label of the same arrow on the child diagram must be identical. (A quality automated tool should not allow this mistake to occur in the first place.)
- Arrow labels must be unique. (It is inappropriate to make the reader interpret the label from the context of the diagram.)



These two rules are machine detectable via character string matching.

We have a couple of minor items from UM 6.4.2:

- Separate labels as far as possible from boxes, corners and intersections.
- If an arrow is long label it twice.

The area in which we encountered the greatest difficulty is in the labeling of arrow branches and joins. The apparent motivation behind the attempt to define arrow segments in the UM was to pave the way for saying, "Every arrow segment must be labeled.". All that was actually said was, "**Arrow labels are associated with arrow segments.**". A consensus of opinion at the IDEF Quality Metrology Working Group in Ft. Worth⁴ was that this statement did not constitute a requirement for a label on every arrow segment. There was no disagreement that direct interfaces should be labeled nor that boundary arrows should be labeled. The concern was relative to the labeling of arrows entering and leaving branches and joins.

A significant part of the problem is the omission from the UM (an oversight) of well-known conventions for the interpretation of unlabeled arrows entering and leaving branches and joins (Figure 20). In words, "An unlabeled arrow leaving a branch is assumed to be everything that was part of the arrow entering the branch.", and, "An unlabeled arrow entering a join is assumed to be everything that is part of the arrow leaving the join.". These conventions are pretty much common knowledge in the IDEF₀/SADT community and are usually a part of training materials. But, they're not in the UM.

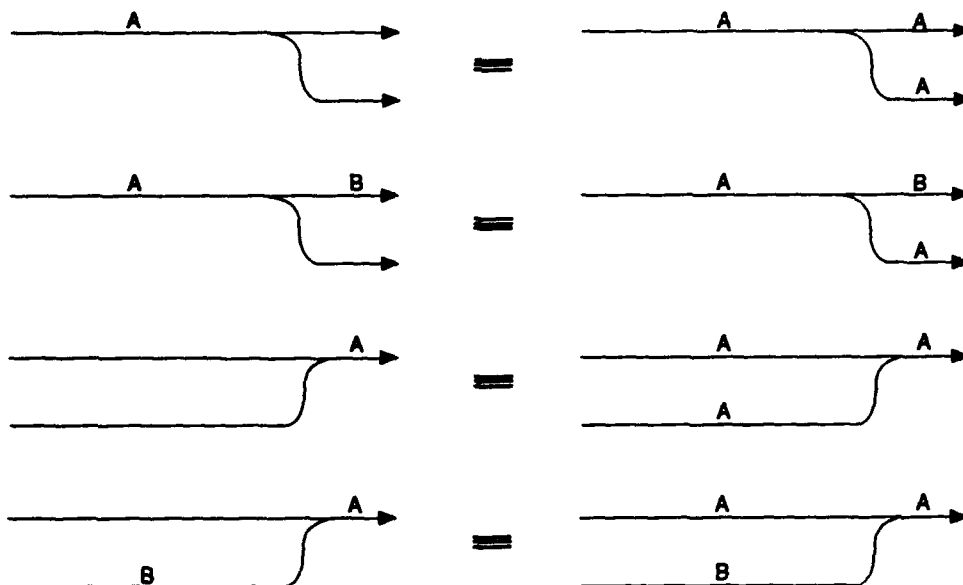


Figure 20. Conventions for Interpretation of Arrows at Branches and Joins

We find the convention for branches useful. It's not at all uncommon for an arrow to branch into carbon copies of itself and go to several places. It's convenient to not have to explicitly label the branches. We consider the convention for joins to be patently absurd. Why would you want to add something to something else which already includes what you're adding before you add it? Why would there be two different sources of "all of the tokens" of the same data? The convention appears to exist to handle things that shouldn't happen in the first place. We suspect this is another Guru rule. We've never seen rationale for it.

So, we have no solid rule requiring labels for all branches. And, we have no conventions (in the UM) for the interpretation of unlabeled arrows. As if that weren't enough of a problem, it turns out that the conventions have a flaw.

The convention (we'll limit the discussion to branching, for convenience) works fine for an arrow which splits into exactly two parts. But when the arrow splits into three, or more, parts we have the problem depicted in Figure 21. The small segment (per

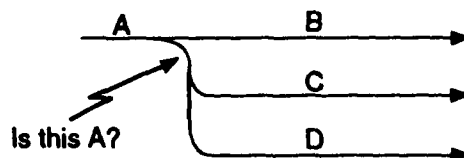


Figure 21. Arrow Segment Ambiguity

UM segment definitions) between two adjacent branches is an unlabeled segment. Thus, by convention, it is A. That means C and D, in combination, are all of A. That, in turn, means B is part of C and D, since B is part of A and the combination of C and D is A. The problem is that this is almost never what the author meant to say. He was simply splitting A into three things; B, C and D. If he intended two consecutive splits into two parts he would have (presumably) drawn it like Figure 22. Is Figure 22 identical, in meaning to

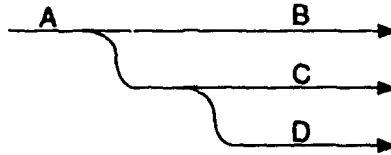


Figure 22. For Comparison with Figure 21

Figure 21? It's probably debatable. What if it had been drawn like Figure 23? (This works for three branches, but not for four, or more, branches.) Figure 23 has no ambiguity.

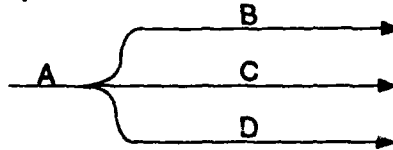


Figure 23. For Comparison with Figures 21 and 22

We have two choices:

- We can dictate that "segments" between consecutive branches do not exist. Figures 21, 22, and 23 mean the same thing --- a "three-branch".
- We can interpret "fan-outs", branches which are aligned perpendicular to the direction of flow of arrows out of the branch (Figure 24), differently from other renderings. Fan-outs are "n-branches" --- the ambiguous segments don't exist. Other renderings are subject to the application of the conventions of Figure 20.

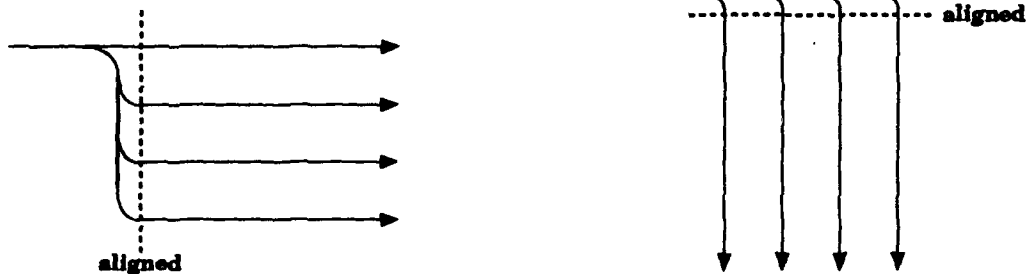


Figure 24. Fan-Outs

So, we have a requirement for formalization. We need to: 1) Define unambiguous conventions for the interpretation of unlabeled arrows, 2) Incorporate the conventions in the UM, and 3) Make an unambiguous statement in the UM regarding the requirements for labeling arrows.

The metric currently treats fan-outs as n-branches and other renderings as subject to the application of the conventions.

3.1.3.3 *Diagram Titles*

The TITLE field on a diagram is governed by one, important, rule. A diagram is the decomposition of a box on the parent diagram. The title of the child diagram is, identically, the verb phrase in the corresponding box on the parent diagram.

3.1.4 *Codes and Numbers*

Codes and numbers includes; rules for box numbering on a diagram; diagram connectivity, or node numbers; data structure connectivity, or ICOM codes; and reference language.

3.1.4.1 *Box Numbering*

Boxes on a diagram are numbered consecutively from one to n, where n is the number of boxes on the diagram. The box number is located on the lower right corner of the box. The order in which the boxes are numbered should indicate the precedence relationships between the boxes --- "1" being the strongest activity, "2" being the second strongest activity, and so on, with "n" being the weakest activity. Frequently, this is simply a numbering from top-left to bottom-right. But, when other than classical diagonal layouts occur, the numbering scheme can be the only indication of precedence relationships.

3.1.4.2 *Diagram Connectivity*

The hierarchical relationship between diagrams in a model is captured by node numbers, the entry in the NODE field of a diagram. The node number consists of the letter "A" and an integer string. The letter "A" is inherited from SADT where the distinction between activity diagrams and data diagrams was necessary. "A" identified a diagram as an activity diagram. In IDEF₀ the "A" is superfluous. The node number of a diagram is exactly one integer longer than the node number of the corresponding parent diagram. The node number of a diagram consists of the node number of the parent diagram with the box number of the box being decomposed appended. For example, the node number of the decomposition of box 3 on diagram A24 is A243.

There are two exceptions, for convenience, to this numbering scheme at the top of the hierarchy. We noted previously that the (implicit) number of the box on the A-O context diagram was zero. This would imply that the node number for the decomposition of the box on the A-O diagram is A-00. The decomposition of the box on the A-O diagram is simply given the node number AO. This diagram is considered to be the top diagram of the model. One would expect the decompositions of the boxes on the AO diagram to have the node numbers A01, A02, ..., A0n. We, in fact, drop the zero and number these diagrams A1, A2, ... , An. All other diagrams in the hierarchy consist of parent diagram node numbers with appropriate parent diagram box numbers appended.

Note that between the node numbers and diagram titles we have a machine implementable consistency, or correctness, check. We can check for a correct diagram title by looking at the verb phrase in the appropriate box on the parent diagram as specified by the node number. Or, conversely, we can find the box on the parent diagram with the same label to confirm that the node number on the child diagram is correct.

Node numbers for FEO's, text and glossaries are identical to the corresponding diagram node number with the character F (FEO), T (text) or G (glossary) appended. An FEO illuminating diagram A245 would have the node number A245F. Additional characters have the meanings you would logically expect them to have. If a glossary for diagram A223 is two pages long you would find A223G1 and A223G2 as node numbers on the glossary. If text were required to illuminate an FEO associated with diagram A162, the node number would be A162FT.

3.1.4.3 Data Structure Connectivity

Data structure connectivity is provided by ICOM codes and parentheses. Technically, ICOM codes provide connectivity and parentheses denote a break in connectivity. Every boundary arrow on a diagram must have an ICOM code, or be enclosed in parentheses, near the margin of the diagram.

ICOM codes identify the correspondence of arrows on the parent diagram with arrows on the child diagram. The ICOM code is redundant since that correspondence should also be discernible from arrow labeling. Thus, the capability for automated cross-checking is available.

The ICOM code for an arrow consists of the letter I (input), C (control), O (output) or M (mechanism) and an integer. The integer denotes the position of the arrow on the box on the parent diagram which is being decomposed. Controls and mechanisms are numbered from left to right. Inputs and outputs are numbered from top to bottom. The ICOM code assignment scheme is shown in Figure 25. These ICOM code assignments are implicit. They are not shown on the parent diagram. Parenthesized arrows, i.e., arrows enclosed in parentheses where they touch the box on the parent diagram, receive an ICOM code assignment even though the arrow will not appear on the child diagram (re: UM Fig. 3-21).

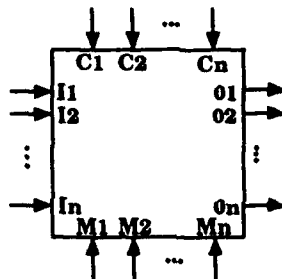


Figure 25. Implicit ICOM Code Assignment on Parent Diagram

Every arrow on the child diagram must have an ICOM code, or be enclosed in parentheses to denote that the arrow does not appear on the parent diagram. Arrows on the child diagram need not be arranged the same as the arrows on the parent diagram box. Arrows may change roles --- a control arrow on the parent diagram box may be an input arrow on the child diagram. The ICOM codes on the child diagram unambiguously identify the correspondence. Figure 26 (UM Fig. 3-18) is an example.

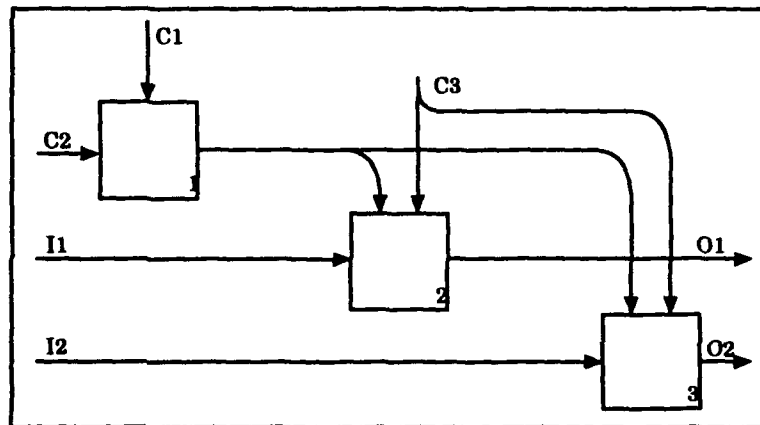
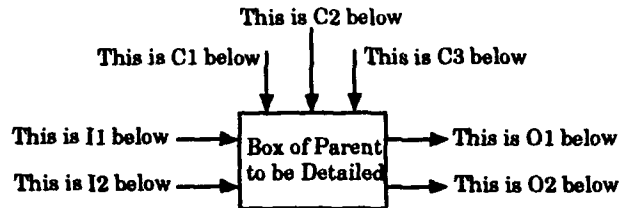


Figure 26. ICOM Coding Example

ICOM coding is not used to denote arrow bundling between parent and child diagrams. Arrows on the parent diagram which receive ICOM code assignments must appear, separately, on the child diagram. An arrow on the child diagram may not have more than one ICOM code (re: UM Fig 6-2).

Metrics for ICOM codes include: checking for the presence of an ICOM code, or parentheses, on the child diagram; checking for format correctness (the letter I, C, O or M and an integer); cross-checking, via arrow labels, to determine the correctness of the ICOM code; and checking to confirm that all non-parenthesized arrows on the parent diagram box are present on the child diagram (i.e., all ICOM codes are accounted for). These metrics are all machine implementable.

We offer a recommendation for a new IDEF₀ construct providing connectivity for tunnelled arrows. We believe an arrow parenthesized at the parent diagram box should be shown at the margin of the child diagram. The arrow should be labeled, should have its assigned ICOM code, and have its arrowhead enclosed in parentheses (Figure 27). This construct accomplishes two things. (Previously) missing ICOM codes are accounted for on the child diagram. Important constraint information (typically of the "to all boxes" variety) is visible on the child diagram.

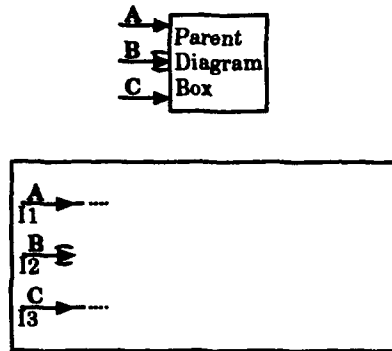


Figure 27. Handling Tunnelled Arrow Connectivity

3.1.4.4 Reference Language

Section 6.5.2 of the UM, titled "References and Notes", describes a reference language for referring to specific parts of diagrams, including diagrams in other models. The use of the reference language is convenient and recommended, but in most instances purely optional.

We are aware of only two situations in which the use of the reference language is required. A downward pointing mechanism arrow is used to indicate that the decomposition of the box is in another model. A reference, consisting of the model name and node number of the decomposition is required. The format is MODEL NAME/NODE NUMBER, e.g., MFG/A21. When a tunnelled arrow tunnels from one place in a model to another place in the model, the reference language should be used to denote the connectivity. However, we consider this to be an improper use of tunnelling, pathological coupling.

With the exception of checks for the presence of the required reference notations noted above we propose no metrics for reference language usage. Usage undoubtedly improves model quality. But, the effectiveness of the usage is solely a human review item.

3.2 Decomposition Quality Factors

Decomposition quality factors is the name we've given to quality indicators which are not, in general, based on syntax nor semantics issues. Sometimes the distinction is subtle. One could argue, for example, that coupling structure is (or should be, with additional rules formalization) a syntax issue. But, generally, decomposition quality factors are measures of model presentation such as amplification, cohesiveness, specificity, complexity and balance.

We discuss decomposition quality in four topics: Coupling Metrics, Cohesion Metrics, Activations, and Balance. These topics go beyond the concept of metric primitives but still qualify as primitive metrics. We elaborate on that distinction in the Algorithm Development presentation in Section 4 of this report.

3.2.1 Coupling Metrics

We discuss coupling from three perspectives: pathological or content coupling; control coupling; and coupling structure. We consider these to be distinct viewpoints on coupling leading to distinct approaches for evaluation or measurement. Only the coupling structure viewpoint leads naturally to a quantitative metric.

3.2.1.1 Pathological Coupling

Pathological coupling occurs when one activity depends upon information from another activity but there is no arrow between the boxes providing that information. An exchange of information is occurring, or has occurred, but is not depicted explicitly in the model. The coupling is implicit, or pathological.

The detection of pathological coupling is a review-by-system-expert function. One must understand the relationship between two activities in order to ascertain that the relationship has not been depicted by an arrow (a missing arrow). We consider this to be a model validity issue.

Although no quantitative measure of pathological coupling is apparent, a "help" function is automatable. A tunnelled arrow from one place in a model to another place in the model is pathological coupling. Any tunnelled arrow suggests the possible existence of pathological coupling in a model and can, thus, earmark an item for system-expert-review attention. The presence of tunnelled arrows is, of course, machine detectable.

3.2.1.2 Control Coupling

Control coupling is similar to pathological coupling in one respect. Control coupling also occurs when one activity is using information about another activity which has not been explicitly provided by an arrow. However, even if the information were provided by an arrow, the model (or possibly the system depicted by the model) would still be wrong. Since control coupling is an error of commission, rather than omission, we consider it to be a more serious error than pathological coupling. However, since control coupling is typically visible in the model when it occurs, it is easier to detect and, thus, easier to correct. So, from a detection and correction viewpoint, control coupling is, ultimately, less of a problem than pathological coupling.

Control coupling occurs when one activity generates data which tells another activity what to do. This requires the sending activity to have knowledge of the internal workings of the receiving activity. Note that an arrow from the receiving activity to the sending activity disclosing the receiving activity's internal workings is not an appropriate solution. That's why we said, "even if the information were provided by an arrow the model would still be wrong". According to Dickover, "Control (coupled) data is not real data being used by the system, but is artificial data created by activities to coordinate incohesive decompositions of the problem. An alternate decomposition always exists that will eliminate the need for control coupling."

Detection of control coupling is a human review (not necessarily by a system expert) function. Control coupling is a decomposition quality indicator requiring arrow label interpretation. Clues are verbs, verb phrases, flags, switches, function selection codes, in the arrow label. Automated detection would require natural language interpretation. (A possible exception is discussed in Section 3.2.2.3.) It is, perhaps, worth noting that the detection of verbs, or verb phrases, would not stress the natural language interpretation state-of-the-art.

3.2.1.3 Coupling Structure Metric

The distinction between environmental or common coupling, external coupling, and record or stamp coupling involves interpretation of the vagueness, or globalness, of arrow labels. However, we believe a valid metric exists which ignores those distinctions. Our premise is that how global a data item is correlates, highly, with how many boxes it goes to.

A coupling structure problem exists, potentially, whenever an arrow splits into carbon copies of itself and goes to several boxes. The occasional occurrence of arrows splitting into a couple of branches, and going to a couple of boxes, is not a severe problem. In fact, many practitioners would argue that that's not a problem at all. Frequent occurrences is likely a problem. A large number of branches, per occurrence, is undoubtedly a problem.

We propose a metric which counts the number of occurrences and the number of branches per occurrence of arrows which split into carbon copies of themselves. We simply refer to this as a Coupling Structure Metric, without alluding to characterizations such as environmental, record, or external.

A fairly simple algorithm would count the number of arrow sinks on a diagram. Then the algorithm would trace back each arrow to see if the arrow came from a branch (on the same diagram). If the arrow came from a branch it would be counted as an arrow with a coupling structure problem. Only unlabeled arrows are considered to be part of the branch insofar as this metric is concerned. An arrow which splits into four arrows, two of which have unique labels, is only considered to be a split into two arrows, the two unlabeled arrows, for the purpose of this metric. In addition, for each occurrence of a branch, one arrow is considered to be legitimate. In other words, an arrow which splits into two carbon copies of itself produces only one "problem" arrow (no need to specify which one), an arrow which splits into three carbon copies of itself produces two "problem" arrows, and so on. This caters to the occurrences vs branches per occurrence consideration. Two occurrences of a split into two arrows (producing four arrows) produces two "problem" arrows. One occurrence of a split into four arrows (again, producing four arrows) produces three "problem" arrows. The "score" for the diagram would be: the number of sinks minus the number of problems, divided by the number of sinks. The idea being a score of 100% if there are no problems, 90% if ten percent of the arrows have problems, and so on.

Note that the metric has a tolerance (unknown, but believed to be acceptable) since some branches are not problems at all. We're not interested in whether the author got a score of 93% vs 97%. We're interested in whether the score is 30%, or 60% --- failing scores, or marginal scores.

The algorithm described represents a preliminary, or candidate, approach. There are other considerations. How do we handle arrows that continue to branch into carbon copies of themselves as we go deeper into the hierarchy? A branch into five arrows on a given diagram may really be a branch into thirty arrows if we trace each one to its ultimate sink in the hierarchy. Simply producing a diagram score may be inadequate.

3.2.1.4 *A Control on Every Box*

The measurement of whether or not every box in a model has a control is a syntactic correctness item. We've chosen to discuss it here because of an intimate relationship to coupling structure problems.

Splitting a control arrow into carbon copies of itself and sending it to the top of several boxes is not the way to provide a control on every box.

Our recommended metric is that each box must have at least one unique control arrow. We say "at least one" because the existence of one satisfies the syntax, even if the box has half-a-dozen other controls which possess coupling structure problems. As with the coupling structure metric, we have the "minus one" consideration. An arrow which branches and goes to several boxes provides a legitimate control to one of those boxes (unspecified).

3.2.2 Cohesion Metrics

In our discussion on cohesion in Section 2.2.2 we noted that although a useful baseline of primitives exists, we felt that seven levels of granularity was overkill and that the simple addition of primitives to get a whole diagram score was invalid. We want to elaborate on both of these issues.

3.2.2.1 Direct Interfaces

We believe that the entire concept of cohesion can be embodied in the existence of direct interfaces between the boxes on a diagram. If there are direct interfaces, then the cohesion is good. If there are not direct interfaces, then the cohesion is poor. But, there are still levels of granularity, based on structure, even when direct interfaces are present, e.g., Figures 9 and 11.

We want a minimum acceptability criterion based on direct interfaces between boxes which, when met, scores 100%. The cohesion may be stronger by virtue of having more direct interfaces than required. But as you add arrows to a diagram you increase its complexity. We don't want a cohesion metric that produces higher and higher cohesion scores as the diagram becomes more and more complex.

The best of all possible worlds is when there is a direct interface path from the first to the last box on a diagram (Figure 28). We don't consider it to be particularly important whether the direct interface is a control, an input or a mechanism (see discussion in Section 2.2.2.2). We would like such a diagram to score 100%. This situation can be described as, "Every box provides a direct interface to the next lower precedence box on the diagram."

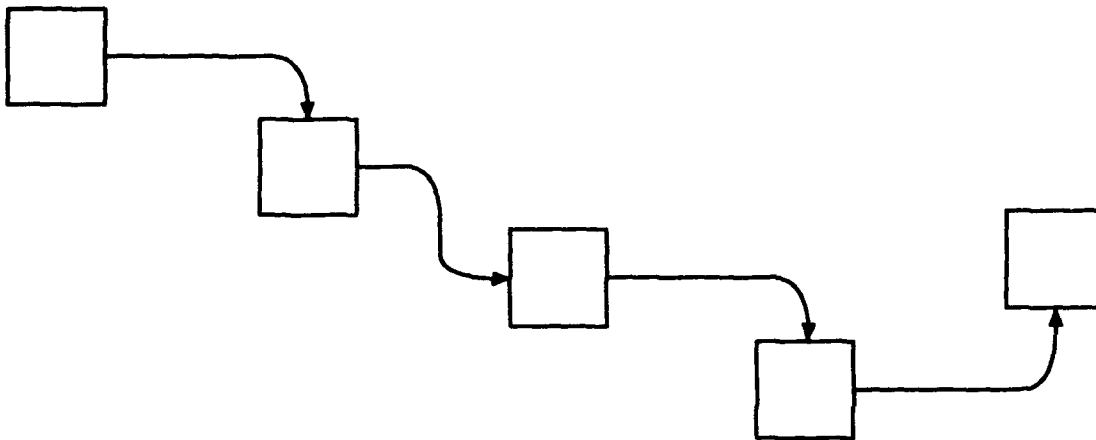


Figure 28. A Direct Interface Path

What if we have two direct interface paths? Figure 29 depicts an example. This is also good cohesion. Intuitively, the Figure 29 cohesion is not as good as the Figure 28 cohesion. "Every box provides a direct interface to a lower precedence box on the diagram." What is the rationale for saying Figure 28 is better than Figure 29? One path is better than two paths.

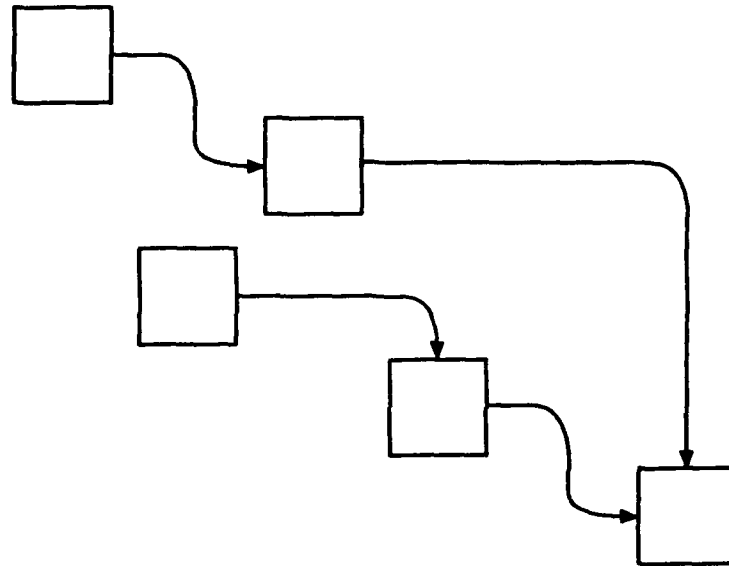


Figure 29. Two Direct Interface Paths

The maximum number of direct interface paths on a diagram is equal to the number of boxes minus one. Figure 30 depicts two examples. In a. all boxes provide a direct interface to the last box. In b. the first box provides a direct interface to all lower precedence boxes. Intuitively, we feel that a., All-to-One Cohesion, is stronger cohesion than b., One-to-All Cohesion. Figure 30 a. is a form of activation cohesion, our name for the procedural cohesion example in Figure 7. Figure 30 b. comes closer to being an example of procedural, temporal or logical cohesion --- boxes with a common control. Although, the direct interfaces needn't be controls. There are many possible structures satisfying "Every box provides a direct interface to a lower precedence box on the diagram." between two direct interface paths and the maximum number of direct interface paths. Indeed, there are several "max" structures between Figure 30 a. and b.

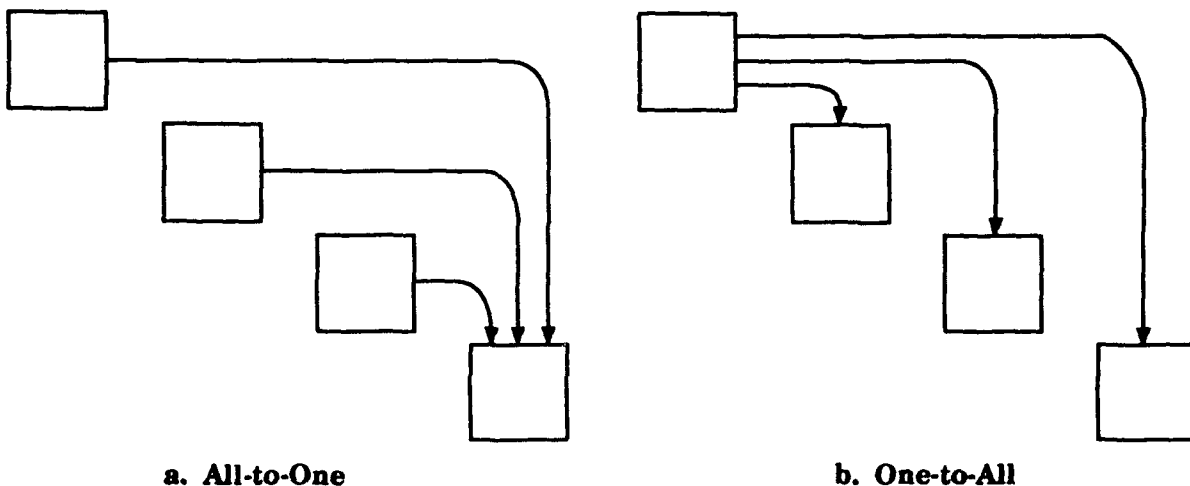


Figure 30. Maximum Number of Direct Interface Paths

Lower diagram cohesion scores involve breaks in the paths, such as in Figure 31. The more breaks, the lower the score. Coincidental cohesion, no direct paths, scores a zero.

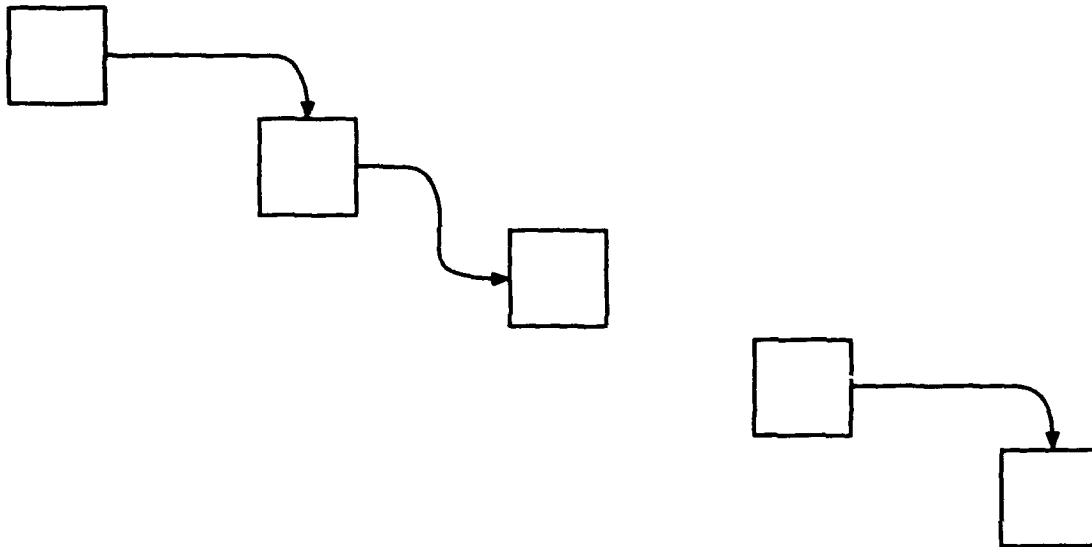


Figure 31. A Broken Direct Interface Path

Clearly, we are a long way from specifying an algorithm for whole diagram cohesion. The development of such an algorithm is recommended future research. But, we believe basing the algorithm on direct interface path structures has high validity. We wanted to introduce the topic in this report in order to stimulate feedback.

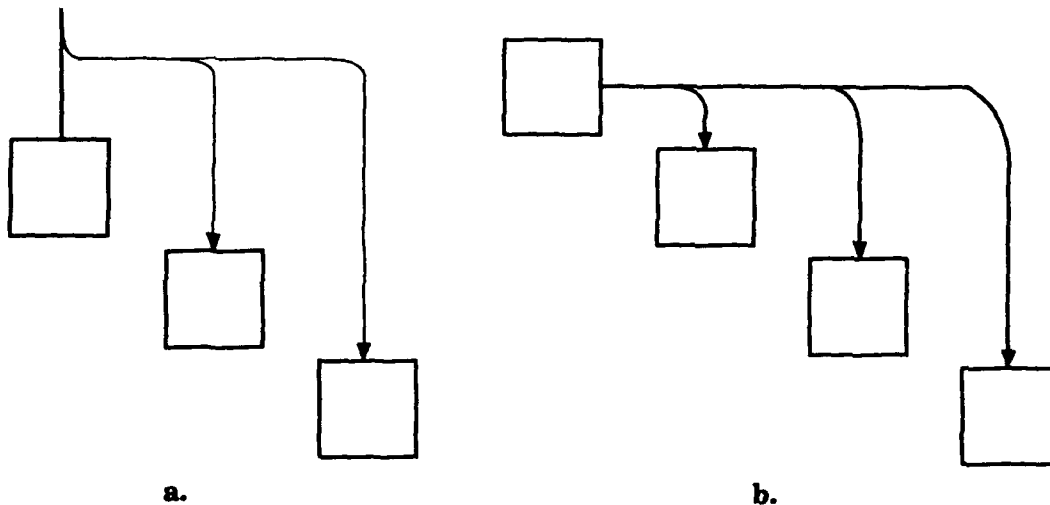
There are a couple of additional considerations worth introducing. One consideration involves a formalization relative to arrow joining. The other consideration is the hierarchical placement of a common source of direct interfaces (e.g., Figure 30 b.) on the same diagram with the sinks vs on the parent diagram.

3.2.2.2 Information Integration

The joining of arrows creates a bundle. As we indicated in Section 3.1.2.2.2.1 there are several motivations for bundling arrows. None of those motivations includes integration of the data! In the software community bundled arrows were frequently viewed as labelled common blocks. Bundled arrows are sets of information in the purest sense --- data items which can be described by a single label, a hierarchical relationship. The integration of information requires an activity, an integration function, a box, with a control dictating the rules governing the integration activity. Consider Figure 8 one more time. What if there had been a sixth box, "Integrate into Manufacturing Plan", instead of just an arrow join? The output would have still been "Manufacturing Plan". And, the ability to branch out components of that plan in other parts of the model is not prohibited by the fact that the information is "properly" integrated. But, look what would have happened to the cohesion of the Figure 8 diagram. Every box would have had a direct interface to a lower precedence box, the hypothetical sixth box. That would have been good cohesion. We believe the community has fallen into the habit, or trap, of treating arrow joins as information integration. One needs only to consider the value of specifying the control on an integration activity, to perceive the value received, in the model, of doing so. (Note that in Figure 8 "Manufacturing Plan" may not be a document. It may simply be a collection of the documents indicated on the diagram. If so, the join is correct. No integration has occurred.)

3.2.2.3 *Inversion of Authority*

In the workshop materials in Albuquerque³ we indicated that environmental coupling with its source in the parent diagram (Figure 32 a.) was worse than the same environmental coupling shown on the child diagram (Figure 32 b.).



**Figure 32. Environmental Coupling Source
Off or On a Diagram**

Based on the recommended direct interface concept of cohesion metrics the rationale is apparent. Figure 32 b. shows the direct interfaces and Figure 32 a. doesn't. Although, technically speaking, there is no difference between the reality of what is being depicted. The difference is just how we show it in the model. The ramifications are far deeper than it appears.

In Dickover¹⁰ there is a discussion on "Inversion of Authority" The discussion is somewhat difficult to follow. But, in a nutshell, what Dickover was saying was that, although a decomposition of box A on a parent diagram into boxes B, C, and D on a child diagram means "B, C and D are A", in a software implementation "A" has code! The code is "CALL B", "CALL C", "CALL D". The implementation of an identity relationship has its own code. We fall into a trap, in functional modeling, when we put box A on the same diagram with boxes B, C and D, to depict an invoking type of relationship. In fact, an invoking control is control coupling. You have to know what the activities are doing in order to correctly invoke them. Thus, in some cases, poorly structured coupling can be an indication of control coupling.

If you accept the Inversion of Authority concept, then our presentation in Albuquerque was incorrect. Whatever the "score" value of cohesion which is provided by poorly structured coupling turns out to be in our algorithm, making a distinction between what is "off of the diagram" and what is "on the diagram" may not be a valid consideration. A functional relationship and an implementation relationship are not the same thing. A parent diagram box may have its own implementation: software calls; wires between electronic modules; racks, nuts and bolts attaching hardware items; physical or geographical proximity of personnel who perform activities. Such items integrate the child diagram boxes in a manner which does not constitute a functional relationship.

3.2.3 *Activations*

Activation theory offers a lot to model quality evaluation. Activations are a measure of complexity. Activation disclosure is a measure of amplification. Minimum activations can provide a formal, quantifiable, measure of whether a system has been decomposed to a useful level of detail.

Formalization is required. The UM describes activations. **"A box may perform various parts of its function under different circumstances, using different combinations of its input and controls and producing different outputs. These are called the different activations of the box."** (UM 3.1.3) But, this description occurs, in passing, in the context of another discussion. No activation theory is presented.

3.2.3.1 *Activation Theory*

Activation theory is conceptually simple. Boxes are activated by a presence of entities (e.g., data, information, objects) on arrows. A box may have several possible activations depending on the presence or absence of information in the various combinations of arrows constraining the box. The required activation theory development is the definition of activation rules which are consistent with the IDEF₀ methodology.

Here's an example. Figure 33 depicts a box with five arrows. Writings on activations say that the number of possible activations provided by five arrows is 2^5 (i.e., 32). The truth table in the figure depicts the basis for saying there are 32 possible activations. Note that just because an activation is possible doesn't mean that an activation actually occurs. The base "2" reflects the two states "presence" or "absence" of data in an arrow. Actually, 32 is far too large a number for the possible activations in Figure 33, in the context of IDEF₀. Let's take a closer look at the truth table. We've numbered the rows, for reference, so we can talk about them. The row numbers have no other significance.

We first note that the first sixteen table entries, i.e., the first sixteen rows, correspond to cases of no data present in the control arrow (zeros in the C1 column). We submit that since the methodology requires a control on every box, a "possible activation" must require the presence of data on at least one control arrow. Thus, within the context of IDEF₀, the first sixteen rows are not possible activations. We'll limit further discussion to rows 17 through 32.

Now, look at rows 17, 21, 25 and 29. These rows correspond to no output data (zeros in both the O1 and O2 columns). Again, consistent with the IDEF₀ rules, we submit that "no output" means "no activation". A "possible activation" must produce data in at least one output arrow.

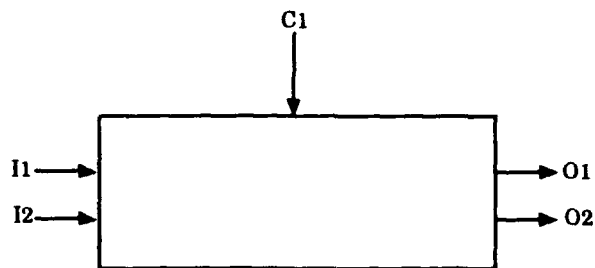
So, in fact, the box in Figure 33 has twelve possible activations, not 32. Note that there is no requirement in the methodology for input arrows. Thus, there is no requirement for data in the input arrows to produce an activation.

The number of possible activations of an IDEF₀ box can be formulated as,

$$\text{No. of possible activations} = 2^I \cdot (2^C - 1) \cdot (2^O - 1) \cdot (M^*)$$

where the exponents are: I = no. of input arrows, C = no. of control arrows, O (alphabetic) = no. of output arrows. Since portrayal of mechanisms is optional in IDEF₀, we have,

$$\begin{aligned} M^* &= 1, \text{ if no mechanism arrows are depicted} \\ M^* &= 2^{M-1}, \text{ if mechanism arrows are depicted} \end{aligned}$$



Reference	C1	I1	I2	O1	O2
1	0	0	0	0	0
2	0	0	0	0	1
3	0	0	0	1	0
4	0	0	0	1	1
5	0	0	1	0	0
6	0	0	1	0	1
7	0	0	1	1	0
8	0	0	1	1	1
9	0	1	0	0	0
10	0	1	0	0	1
11	0	1	0	1	0
12	0	1	0	1	1
13	0	1	1	0	0
14	0	1	1	0	1
15	0	1	1	1	0
16	0	1	1	1	1
17	1	0	0	0	0
18	1	0	0	0	1
19	1	0	0	1	0
20	1	0	0	1	1
21	1	0	1	0	0
22	1	0	1	0	1
23	1	0	1	1	0
24	1	0	1	1	1
25	1	1	0	0	0
26	1	1	0	0	1
27	1	1	0	1	0
28	1	1	0	1	1
29	1	1	1	0	0
30	1	1	1	0	1
31	1	1	1	1	0
32	1	1	1	1	1

Figure 33. Box Activations and Truth Table

The exponent M is the number of mechanism arrows, and the formulation incorporates a requirement for the presence of a mechanism entity in at least one mechanism arrow. For our example we have,

$$\text{No. of possible activations} = 4 \cdot 1 \cdot 3 \cdot 1 = 12$$

We make no claim that this is all there is to activation theory. But, we wanted to make the point about consistency with the rules of the methodology.

We consider the number of possible activations of boxes on diagrams to be a valid measure of diagram complexity. This provides a useful relative measure, which includes quantifiability. And, it is machine detectable and computable. We can say "Diagram 1 is three times as complex as Diagram 3.", or "... five times as complex", or "... 50% more complex". An absolute metric, on the other hand, requires a standard or rating scale. Such a standard would have to evolve over a period of time, based on recommendations and feedback from the IDEF₀ community. We note that, "no more than four arrows per side of a box", allows possible activation numbers as high as 54,000 per box. A rule like "no more than six arrows per box" limits possible activations to around 30 per box (and that's if all six arrows are non-mechanisms). Activation theory may provide the needed rationale for an "arrows per box" syntax rule.

3.2.3.2 *Activation Disclosure*

Activation disclosure, i.e., elaboration of possible activations of a box in the decomposition of the box, has potential as a quantifiable, and machine detectable, measure of amplification. Assume Figure 34 is the decomposition of the box in Figure 33.

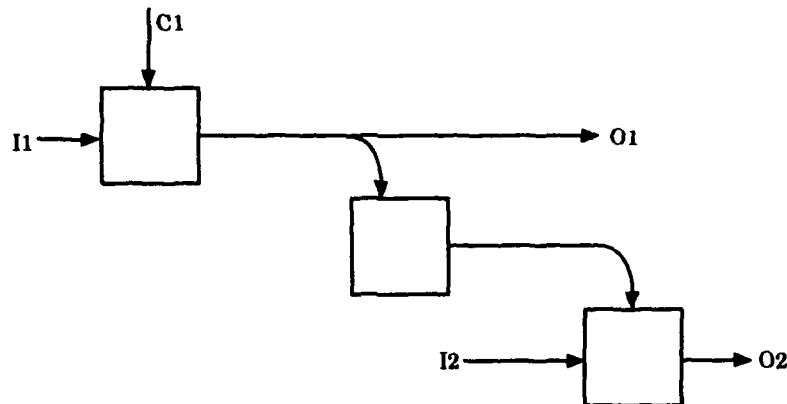


Figure 34. Decomposition of Box from Figure 33

We note that I2 cannot produce O1, I2 provides no constraint impacting O1. This observation eliminates rows 23, 24, 31 and 32 from the table in Figure 33 (rows containing 1's in both the I2 and O1 column). Now, we have eight possible activations instead of twelve.

Note that relative quantifiability is all we need for an amplification measure. The decomposition has reduced the number of possible activations by one-third. It's not obvious how you can convert an "uncertainty reduction" number into an "amplification" number. Have we amplified our knowledge by 33%, by specifying the "truth" about four of the 12 possible activations? There are additional factors which need to be researched.

Tunnelled arrows can cause a child diagram to have more possible activations than the parent box. This consideration may provide rationale for more specific tunnelled arrow usage rules, i.e., formalizations.

Intuitively, it appears that feedback arrows on a child diagram may cause a child diagram to have more possible activations than the parent box. We haven't pursued this. This consideration may propagate usage rules for feedback in IDEF₀ diagrams.

Dickover¹⁰ states that feedback arrows are undesirable and can usually be eliminated by restructuring the decomposition. That viewpoint may be unique to the software module independence scenario. Software modules don't negotiate outcomes. Enterprise activities do --- frequently.

3.2.3.3 *Minimum Activations*

The minimum number of possible activations an IDEF₀ box can have is either one or two: exactly one control arrow, exactly one output arrow, exactly one mechanism arrow (whether shown or not), and at most one input arrow. This produces one possible activation if there is no input arrow and two possible activations if there is an input arrow. Note that both kinds of minimum activations will always exist in a model. The only way one can reduce a diagram to boxes with only controls is to tunnel away (black hole variety) existing input arrows.

A diagram consisting solely of minimum activations on each box (e.g., Figure 34) is not necessarily decomposed as far as possible (useful). But, minimum activation diagrams may provide a quantifiable, and machine detectable, measure of how deep a decomposition is "required" to go. This concept warrants further research and IDEF₀ community feedback.

3.2.3.4 *Activation Language*

Marca⁹ describes an activation language, or activation notation, from SADT, which allows authors to specify the activations of a box on an SADT diagram. These notations would be a valuable addition, even as an option, to IDEF₀. The notations are machine interpretable.

3.2.4 *Balance*

There are at least four balance considerations which provide measures of decomposition quality. We introduced these in Section 2.1.2.3. We elaborate on them here.

3.2.4.1 *Balance Between Box Detail and Arrow Detail on a Diagram*

The UM stated that a diagram with very abstractly labeled boxes but minutely labeled arrows is semantically unbalanced. If you believe the Doug Ross quote in Section 3.1.3.1, this semantic unbalance is not a problem.

However, the converse situation in which the boxes are well-labeled but the arrows have vague, generic, labels is a problem. Notwithstanding Mr. Ross's comments, we would like to have good labels on both boxes and arrows. And, in fact, arrows are supposed to be decomposed in the hierarchy, too. We don't just decompose boxes. We have a right to expect the decomposition of a vaguely labeled arrow to disclose its meaning.

When that disclosure doesn't occur, what has happened? We have the coupling structure problem discussed in Section 3.2.1.3. Although this form of semantic unbalance may be an appropriate human review checklist item, we believe the coupling structure metric will always detect the problem as a failure to decompose the arrow. In other words, an automated system may fail to detect the problem on the diagram where it first occurs, but will catch it later as a decomposition glitch. So, we propose no specific metric for this balance problem.

3.2.4.2 *Balanced Diagram Level of Detail*

One would expect diagrams at the same level in a hierarchical decomposition to depict similar levels of detail. We have a somewhat unique consideration here. We have an unbalance condition which is very easy to measure, to quantify. All we have to do is compare the number of boxes per diagram and the number of arrows per box on diagrams

at the same level in a decomposition. But, how much do we care? How important is this balance, or unbalance, issue?

So, for a change, our problem isn't measurability. Our problem is how we rank the problem, once measured, in the overall scheme of things. Intuitively, small variations are of little, or no, concern. Large variations probably indicate a decomposition strategy problem. In a structured team-modeling environment, large variations might indicate a difference in the skill levels of the authors. This would be a useful management metric --- alerting the team leader to skill differences of team authors. We solicit IDEF₀ community feedback on the utility of this metric.

3.2.4.3 *Balanced Level of Decomposition*

The concern here is when some boxes on the A0 diagram are decomposed to six levels of detail and other boxes on the A0 diagram receive little or no decomposition. We are concerned with wide variations in level of decomposition. We are not concerned with small variations. One part of a system may just be more complex than another part. So, the model reflects that difference in complexity.

However, we can't lose sight of the fact that, in the model, our decomposition strategy has defined the "parts". Variations in complexity may be the result of our decomposition strategy rather than a characteristic of the system being analyzed.

In extreme cases, this unbalance suggests a redefinition of context. If only one box on the A0 diagram has been decomposed, then the A0 diagram should become an A-1 diagram, and the decomposed box should become the "model", i.e., the A-0 box.

As was the case with Balanced Diagram Level of Detail, we have something which is easy to measure, and the metric is easy to automate. The problem is ranking the measurement in the overall scheme of things. We solicit IDEF₀ community feedback on the utility of this metric.

3.2.4.4 *Arrow Bundle Join and Branch Detail*

The level of detail of arrows joining to form a bundle and the level of detail of arrows branching out of a bundle should be balanced, i.e., similar, or the same. Figure 19, in Section 3.1.2.2.2.1, depicts the extreme case in which a "consistency" problem steps over-the-line and exhibits itself as a "correctness" problem. We are concerned here with consistency, balanced level of detail, rather than correctness.

The balance problem occurs only with bundled arrows which are both created, by joins, and used, by branching in the model. If a bundled arrow enters the model, from outside, and is decomposed by branching, that's an ideal state of affairs. No consistency problem exists. If a bundled arrow is created, by joins, and leaves the model as a bundled arrow, that too is an ideal state of affairs. No consistency problem exists. This is simply the decomposition, in the model, of an output arrow.

The balance problem occurs when an arrow, created by joins, goes to other boxes in the model without being decomposed. Often, the coupling structure problem exists when this happens. The converse situation occurs when an arrow is created as the output of an activity as a bundled arrow, without disclosure of component parts via a join, and is later decomposed, by branching, into component parts and routed to several activities. The Manufacturing Plan arrow in the Manufacture Product model exhibits both of these balance problems (as well as the Figure 19 problem).

The significance of this balance problem is related to cohesion. A component arrow on diagram "A", which is joined into a bundle, and the same component arrow which branches out to a box on diagram "B", is a direct interface arrow from the source box on diagram "A" to the sink box on diagram "B". The two boxes have strong cohesion, but are

on different diagrams. Maybe they belong on the same diagram (via a different decomposition) and maybe they don't. But, if the unbalance problem exists, the cohesion is not disclosed at all. We believe that a lot of clues to an improved decomposition are clouded, if not buried, by unbalanced arrow bundling. The author, using unbalanced arrow bundling, has hidden, from himself, possible alternative decompositions.

This metric, automatable by algorithms for arrow tracing, is considered to be a significant model quality measure. In particular, the metric is a measure of the extent to which an author has disclosed, to himself, the available decomposition options.

4.0 RECOMMENDED FUTURE RESEARCH

Our purpose in performing the research reported herein was to establish feasibility of substantive IDEF₀ quality metrics. We began with the baseline of IDEF₀ Quality Metrics described in the UM. We knew before we began that this provided a starting point, but fell short of being a useful baseline. The coupling and cohesion metrics qualified as substantive. We found a number of problems in the syntax and semantics material.

Our point is that feasibility for quality metrics existed before we began. Our research was performed to establish the feasibility of substantive quality metrics that, in some significant way, evaluates the quality of the analysis reflected in a delivered IDEF₀ model. Not just syntax and semantics, but decomposition quality factors. The real baseline was Dickover's coupling and cohesion writings.

We refer to our results as metric primitives, the ability to measure isolated substantive factors. There is significant additional research required to integrate the metric primitives into primitive metrics, whole diagram and whole model measures of quality factors. And, the integration of primitive metrics into whole model quality ratings.

Recommended future research outlines the remaining work to be accomplished. The discussion is focused on two issues: Formalizations and Algorithm Development. We very briefly discuss Training Materials Development and Software Specifications.

4.1 *Formalizations*

Formalizations, as used herein, simply refers to an attempt to more precisely define the rules for IDEF₀. A strong motivation in our development of quality metrics was to leave an audit trail. We didn't want our metrics to merely reflect someone's opinion. We wanted our metrics to be traceable to IDEF₀ rules. We had some difficulty in leaving the desired audit trail because we encountered mistakes, omissions, contradictions and Guru rules (rules without published rationale) in the UM. We commented on these as they occurred in our presentation in Sections 2.0 and 3.0 of this report. We summarize the more important observations in this section.

A couple of philosophical issues are worth mentioning. One related to the extent to which well-defined rules are required to support quality metrics. The other relates to the desirability of having well-defined rules in the first place.

Quality metrics are not necessarily based directly on rules. The cohesion metrics are an excellent example. None of the cohesion primitives are based directly on rules. There is no semantic rule that says a control interface is stronger than an input interface, although many of the construction rules say that implicitly. There is no rule that says decomposition by type is wrong. A good practice rule simply says one way of doing something is better than another way of doing the same thing. We based our control-on-every-box metric on whether or not the control was unique to the box in question rather than shared with several other boxes. There is no IDEF₀ rule stating that controls should be unique. We based our metric on the premise that models rich in unique controls are of higher quality than models which utilize widely shared controls. So, the development of quality metrics doesn't depend strongly on rule formalization. The recommended formalizations in this section are considered to be exceptions --- rules which are needed to support quality metric development.

The other philosophical issue is the matter of trying to define the fine line between necessary rules and rules which unnecessarily restrain the analysts' freedom of expression. IDEF₀ provides *structure without restraint*. The structure is applied to the product rather than to the modus operandi, is embodied in simple boxes and arrows, and allows the use of natural language. We believe this structure without restraint characteristic is the primary reason for IDEF₀ popularity. We don't want formalizations to change that.

We identified seven formalizations which we consider to be enhancements to IDEF₀ as presently defined in the UM: Arrow labeling requirements, Interpretation of unlabeled arrows, Data content of arrows, Mechanism usage, Tunnelled arrow usage, Activation definitions, and Generic interfaces, particularly controls, between boxes.

4.1.1 *Arrow Labeling Requirements*

We noted in Section 3.1.3.2 that defining arrow segments and then simply stating that labels are associated with segments is not generally interpreted by the community as a requirement for a label on every arrow segment. In addition, we didn't care for the attempt to define arrow segments in the first place (see Section 2.1.1.2.1). So, we recommend two formalization concepts here: a different way of describing arrows, or parts of arrows, which appear on diagrams, and a requirement for labeling arrows, or parts of arrows. The labeling rules embody the arrow description rules.

- Every direct interface arrow, i.e., an arrow from the output side of a box to the input, control or mechanism side of a box on the same diagram requires a label.
- Every boundary arrow, i.e., an arrow arriving at the boundary of a diagram as an input, control, mechanism (ICOM coded or tunnelled) arrow and going to the input, control or mechanism side of a box on the diagram, or an arrow from the output side of a box on the diagram and leaving the diagram boundary as an output (ICOM coded or tunnelled) arrow, requires a label.
- Every arrow entering a join and the bundled arrow leaving the join require labels.
- The bundled arrow entering a branch and the arrows emanating from the branch require labels.

We recommend making the distinction, described in Section 3.1.3.2, between fan-outs and other renderings of arrow branches and joins in order to clearly indicate that there is no requirement for a label on the "apparent segment" between consecutive branches and joins in a fan-out.

We solicit community feedback on these recommended rules, and on the wording thereof. Note that the following Section 4.1.2 offers exceptions to the rules.

4.1.2 *Interpretation of Unlabeled Arrows*

We observed in Section 3.1.3.2 that conventions for the interpretation of unlabeled arrows emanating from branches and entering joins had been omitted from the UM. The conventions are common knowledge in the IDEF₀ community and exist in training materials.

We recommend incorporation of the branching convention. We find little utility in the joining convention.

A related issue, possibly worth a separate section here in the formalizations material, is more formal definition of rationale for bundling arrows in the first place. The UM, and other writings, simply "allow" bundling. There is no guidance for good practice. We discussed this in Section 3.2.4.4. Rules for branching and joining, and in particular, rules for the interpretation of unlabeled arrows associated with branches and joins, can be based on set theory. That would probably be acceptable to the technical community. But it would probably not be acceptable to the non-technical community.

4.1.3 *Data Content of Arrows*

An arrow is assumed to contain everything that is specified by its label. What that might mean in any particular instance is, of course, subject to the interpretation of the label. However, as we pointed out in Section 3.2.4.1 the content of arrows (and boxes) should be disclosed by their decomposition. This is the underlying concept of Removing the Limitations of Natural Language⁸.

Whatever the content of an arrow, when that arrow constrains a box the box is assumed to require all of the content of the arrow for that constraint. This is the "all of the tokens" concept. This concept provides the rationale for our coupling structure metric.

However, the "all of the tokens" concept is not currently an IDEF₀ semantics rule. The concept was the basis for the environmental and record coupling metrics in the UM. So, it existed as a good practice rule. We feel more strongly about the issue. We believe this should be an IDEF₀ semantics rule.

A statement of the rule sounds like a labeling requirement. "The label on any arrow constraining a box should be indicative of the content of the arrow which is actually used by the constraint. A more generic label indicative of an arrow content greater than what the box actually uses is inappropriate." Coming up with such a label is seldom easy. But coming up with wonderful labels is not as important as the mind-set imposed by the rule. If authors merely think about arrows as all of the tokens required by an activity, then coupling structure problems will be significantly reduced.

4.1.4 *Mechanism Usage*

A mechanism shows how an activity is performed. Every activity requires a mechanism. The activity can't be performed without it. But, we don't have to show mechanisms on our diagram unless it suits us to do so --- unless it's "part of our story".

Is the same argument valid for controls? Have you ever wanted to decompose a system from the "how" viewpoint, and wished you could omit controls as "not part of your story"? We're not advocating that. Just wanted to "run it up the flagpole ...". It's not absurd in an enterprise model where we frequently encounter controls such as "understanding", "experience", "training" and where we're trying to depict "who's responsible" or whether a function should be "manual" or "automated", i.e., mechanism issues.

In any event, we need some formalization on mechanism usage. Mechanism usage, like branching and joining arrows, is "allowed". But, we have no rules for when it's ok, or not ok, or recommended, or required. We don't know whether mechanism cohesion is stronger, weaker, or about the same, as control or input cohesion.

We offer no specific recommendations. Our cohesion metric makes no distinction between control, input and mechanism direct interfaces. The subject warrants some research.

4.1.5 *Tunnelled Arrow Usage*

Tunnelled arrows are another example of something which is "allowed" but for which we have no usage rules. We offered our recommendations in Section 3.1.2.2.2.2, including a new syntax item.

4.1.6 *Activation Definitions*

We believe formal definition of activations of boxes, including an activation notation, or language, would be a valuable addition to IDEF₀. Even without its inclusion, formally, in IDEF₀, we are comfortable with basing complexity and amplification metrics on activation concepts. After all, module (activity) independence is not a formal part of IDEF₀ and that's the basis for cohesion metrics.

Since we devoted an entire section to Activations (Section 3.2.3) we won't repeat that material here, except to restate that there is a great deal more to activation theory than what we included in our discussion. An example, not related to metrics, is the possibility of the machine generation of a process description directly from an IDEF₀ model via activation interpretation.

4.1.7 *Generic Interfaces*

Generic interfaces occur when an arrow branches into carbon copies of itself and goes to several boxes. This is generally undesirable. The Data Content of Arrows discussion in Section 4.1.3 addresses the generic interface problem indirectly. The problem can also be addressed by specific syntax rules. We have mixed feelings about this. We solicit community feedback.

The general form of a syntax rule limiting generic interfaces would be something like, "An arrow cannot branch into carbon copies of itself and go to more than N boxes." Picking N is not trivial. And, it should be based on some substantive rationale rather than be propagated as a Guru rule. We believe N is probably different for inputs, controls and mechanisms. We've already expressed our opinion relative to controls --- every box should have at least one control for which N=1. We believe N may be different depending upon where a branch occurs in the hierarchy. N should refer to the ultimate number of boxes to which the arrow goes in the hierarchical decomposition, not a "per diagram" number.

Our preference is to try the "all the tokens" concept and see if it solves the generic interfaces problem. Then, if a need still exists, consider a syntax rule.

4.2 *Algorithm Development*

The purpose of the research reported herein was to establish the feasibility of IDEF₀ quality metrics. We believe the large quantity of metric primitives identified establishes such feasibility.

However, there is still a lot of work to do in order to develop primitive metrics, the combination of metric primitives into whole diagram metrics and whole model metrics. Primitive metrics are desired by class, i.e., a whole model syntax metric, a whole model semantics metric, whole model coupling and cohesion metrics, whole model balance or consistency metrics, and activation based metrics for complexity, amplification and model completeness. These primitive metrics must then be combined into an overall measure of model quality. We refer to this as algorithm development because we believe all, or most, of it is automatable.

4.2.1 *Syntax Metric Algorithm*

The syntax metric primitives are easily detected. But, the individual detections need to be prioritized, weighted, and combined into a total syntax quality measure. We say prioritized and weighted because some syntax primitives are, intuitively, more important than others.

An important trade-off exists relative to scoring vs alerting an author to errors. Frequently, when a syntax error occurs the automated system knows what was correct. It makes little sense to "score a demerit" when the error can be prevented "on-line". This deviates from a metrics concept. We believe the value of automated "help" exceeds the value of metrology scoring. And, of course, automated "help" doesn't require a syntax error combining algorithm.

4.2.2 *Semantic Metrics Algorithm*

The semantic metrics as presented in the UM received the bulk of our criticism in Section 2. This is the area in which most of the formalism issues arose. Many considerations of what is right and what is wrong are ill-defined. And, more to the point for metrics development, is the problem of defining graduated levels between the extremes of right and wrong. Semantic shortcomings must be prioritized and weighted, including consideration of the distinction between absolute rules, good practice rules, and Guru rules.

The formalization issues should be resolved before the semantic error combining algorithm can be developed. However, resolution can consist of informal community feedback. This is adequate to press on. Formal resolution could take years. The metric will evolve along with the formalizations.

4.2.3 *Coupling and Cohesion Algorithms*

Algorithms are required for the assessment of coupling structure, cohesion, and the rule that "Every box must have a control.". In addition to separate algorithms for these factors, a combined algorithm is required to capture intimate relationships between the factors. For example, some coupling and cohesion measures involve trade-offs between good coupling and good cohesion. Some of the coupling and cohesion factors are impacted by formalization issues. Arrow labeling requirements, arrow data content definitions, generic interface rules and mechanism usage all have an impact.

We presented our preliminary thoughts on these algorithms in the Decomposition Quality Factors discussion, Sections 3.2.1.3, 3.2.1.4 and 3.2.2.1.

4.2.4 *Balance Algorithms*

Balance algorithms, key model consistency measures, were discussed in Section 3.2.4. An algorithm for balanced diagram level of detail (Section 3.2.4.2) is simple but of questionable utility. Balanced level of decomposition (Section 3.2.4.3) is easily measured by node diagram analysis. Utility of this measure is also an issue. Arrow bundle join and branch detail requires an arrow tracing algorithm. This is conceptually simple, i.e., easy for a human, but could be complicated to implement in software for an automated system.

4.2.5 *Activation Based Algorithms*

The basic computation of the number of possible activations of a box was presented in Section 3.2.3.1. Activation disclosure in a decomposition is another example of a determination easily made by a human, but possibly difficult to automate. Path tracing, through nodes (boxes) --- not just arrow tracing, is required to determine whether a

particular input, control or mechanism constrains a particular output. The detection of minimum activations on each box in a diagram is straightforward.

4.2.6 *Total Model Quality Score*

In addition to the algorithms discussed above, we need an algorithm to combine the results into a total model quality score. The algorithm is more complicated than, for example, simple weighted averaging. The metrics are interrelated. We mentioned, in Section 4.2.3, the existence of relationships between coupling and cohesion. Another example is the fact that cohesion and activation disclosure are both measures of amplification. Coupling structure and possible activations are both measures of complexity. Many of the syntax rules for clutter reduction are complexity measures. This is not a trivial algorithm.

4.3 *Training Materials Development*

This was discussed in Section 1.1.5. The key issue is formalization feedback. Many Guru rules occur in training materials in the form of opinions expressed by the instructor or author. We want to avoid the proliferation of Guru rules.

4.4 *Software Specifications*

We have frequently made reference to measures which were machine detectable, machine implementable, machine computable. In most cases we assumed this to be intuitively apparent. In some cases we alluded to how we would do it. But, there is a wide gap between recognizing that something can be automated and writing a rigorous, unambiguous, explicit, specification of how to do it. This is a significant development endeavor.